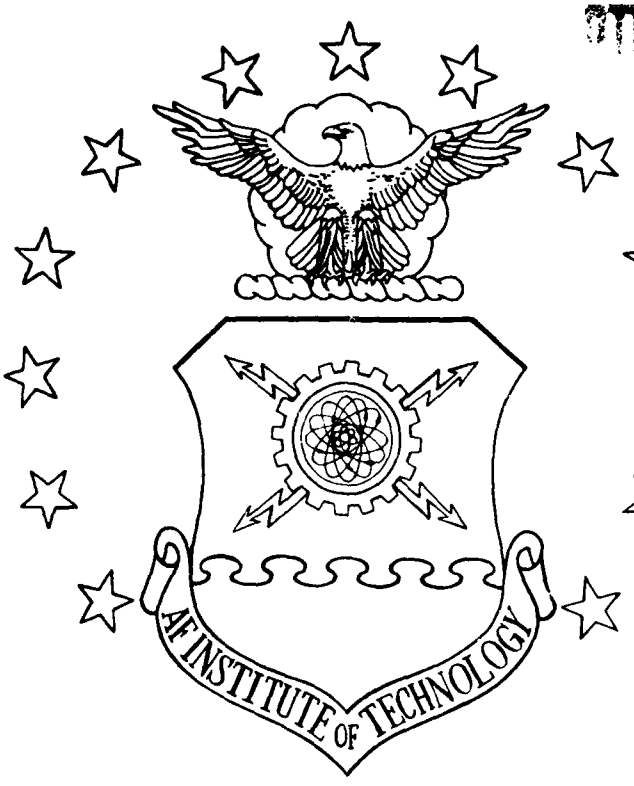
(1)



DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GEO/ENG/89D-1

GABOR TRANSFORMS

FOR

FORWARD LOOKING INFRARED

IMAGE SEGMENTATION

THESIS

Kevin W. Ayer
Captain, USAF

AFIT/GEO/ENG/89D-1

Approved for public release; distribution unlimited

AFIT/GEO/ENG/89D-1

GABOR TRANSFORMS
FOR
FORWARD LOOKING INFRARED
IMAGE SEGMENTATION

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Kevin W. Ayer, B.S., B. E. E.

Captain, USAF

December 1989

Table of Contents

ACKNOWLEDGEMENTS

I express my deepest appreciation and gratitude to Major Steve Rogers, Dr. Matthew Kabrisky, Dr. Mark Oxley, and Captain Robert Williams for their guidance and insight without which I'd surely be gyre and gimbling in the wabe.

I give a very special thanks to Mike Roggemann for his immeasurable help in getting the VAX system to jump through hoops for me and to Dennis Ruck who helped me make pretty pictures out of screen displays and running all the neural network gadgets.

The list is endless, my prose is not,
so I best not be thanking the entire lot.

Yet three more I wish to give thanks with a grin,
to my son, to my daughter, and to my wife, Lynn.

Three people so very important to me,
They waited, and waited impatiently.

Now the journey is over, remember this quip,
AFIT is truly ***Another Fine Intellectual Trip***.

With no regrets, Thank you AFIT.

List of Figures

List of Tables

List of Equations

Abstract

This research investigated the difficult task of segmenting targets in cluttered FLIR images. This research initiated investigation into classifying the segmented targets based on the same method used for segmentation. The primary means for segmenting and classifying targets was the biologically motivated use of Gabor transforms.

This research successfully produced a complete segmentation system based on correlations between FLIR images and non-self similar or modified self similar Gabor functions.

Initial investigation into methods for recognition of targets and their detailed sub-structures was accomplished using self similar and modified self similar Gabor transform correlation coefficients "stacked" in *jets*. Nearest neighbor recognition using the jets method produced accuracy of 0.622. Nearest neighbor recognition of targets using jets for input to a k-nearest neighbor network produced final accuracy of 0.6214 (for calculations based on seven nearest neighbors). An optical Gabor transform design is introduced to implement the computation of the Gabor coefficients.

I. Introduction

The United States Air Force is actively pursuing research in image processing in an effort to reliably detect, recognize, and classify targets. However, a fundamental problem with target detection is segmentation or finding possible targets in highly cluttered images.

The concept of separating one object from another is as common to humans as breathing. We constantly manipulate our environment to suit our needs, whether we physically change the environment (actually move or remove objects within our environment) or simply change the way we perceive or observe the environment.

Often, we cannot physically alter the environment in which a target resides, short of destroying the target and its surroundings. Therefore, we attempt to observe the target in its surroundings. We must separate the target from the rest of the image to observe or detect the target. Distinguishing targets from background is referred to as segmentation.

Gabor filtering and its relevance to segmentation. One method for segmenting images uses texture as the discriminating feature. Since Gabor filtering provides energy localization coupled with textural discrimination, it will be investigated as a biologically motivated alternative to heuristic segmentation schemes. There is

evidence which suggests that the visual cortex of cats react to stimuli much the same way as Gabor functions, or "Gabor-like" functions process information (12:1233-1258).

The Gabor function is comprised of two cofactors. The first cofactor is the Gaussian envelope. The Gaussian envelope provides localization of correlated energy. The second cofactor is the sinusoidal variation over which the Gaussian cofactor is superimposed. The sinusoidal cofactor provides the textural, or frequency, discrimination.

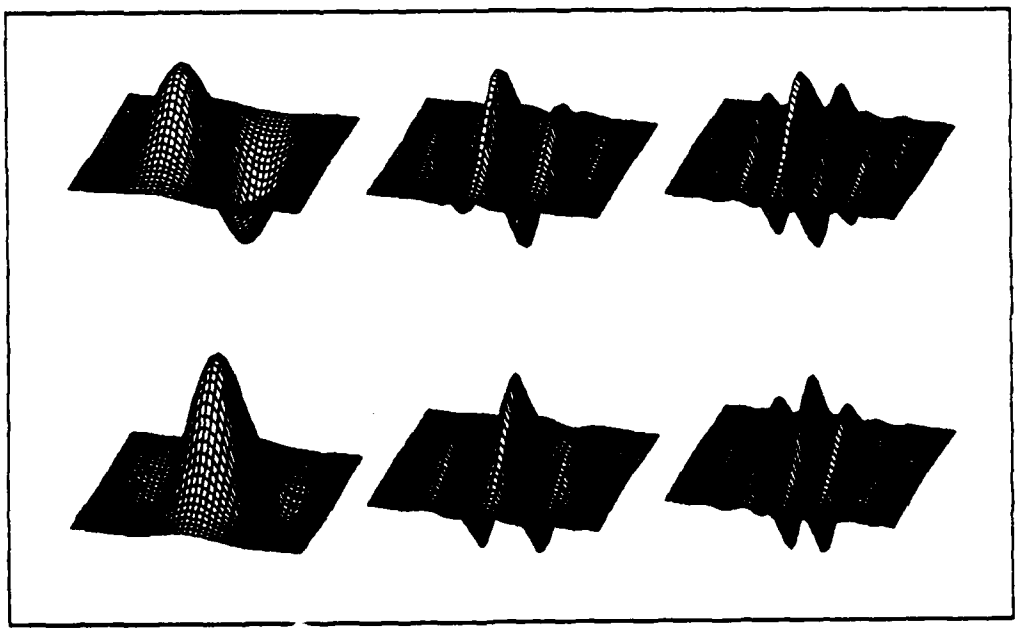


Figure 1.1. Non-self similar Gabor functions. The top row are sine Gabor functions. The bottom row are cosine Gabor functions.

If the sinusoidal variation is independent of the Gaussian standard deviation, the Gabor function is said to be non-self similar (Figure 1.1). Figure 1.1 shows one, three, and five cycle non-self similar sine and cosine Gabor functions.

If the sinusoidal variation is a function of the Gaussian standard deviation, the Gabor function is said to be self similar (Figure 1.2). Figure 1.2 shows sine and cosine self similar Gabor functions possessing one, three, and five standard deviations at the N/2 point.

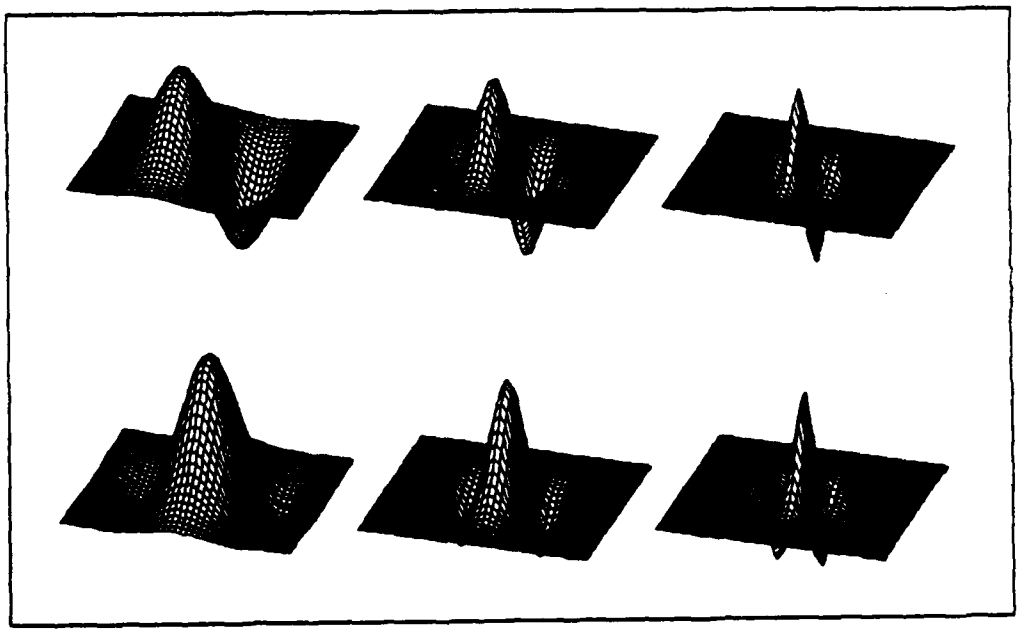


Figure 1.2. Self similar Gabor functions. The top row are sine Gabor functions. The bottom row are cosine Gabor functions.

When the sinusoidal variation is allowed to rotate in-plane with respect to the radially symmetric Gaussian envelope, the Gabor function becomes more specific in its "matched" filter characteristics. Specific spatial frequencies and spatial orientations are chosen to "match" expected target characteristics.

Key Hypothesis. The fundamental hypothesis of this research is that Fourier transform theory is applicable to image segmentation, thus restricting the analysis to linear image processing techniques.

Several heuristic approaches have been taken to effectively segment targets from background (2:MS Thesis, 11:MS Thesis, 18:280-286, 20:18 pages). These heuristic approaches use non-linear mathematical manipulation of data which are not readily accomplished optically; hence, these approaches are, invariably, computationally intense. If Fourier transformation, or an appropriate analog of the Fourier transformation, is a viable approach, the implementation of Gabor filtering using linear Fourier optic techniques should be possible to offset the intense computational effort.

Scope. The first major goal of this research is to develop the simulation code for generating Gabor filters. The second goal of this research is to analyze the Gabor filter for its ability to segment Forward Looking InfraRed (FLIR) images. From the results of the segmentation, the last goal of this research is to provide

a preliminary feasibility study on the use of Gabor coefficients as a means to classify FLIR targets. The requirements necessary for optically generating Gabor correlations are also investigated.

This research begins with a study into the fundamental understanding of time based self similar Gabor transforms for dilations corresponding to near octave frequencies of 53.89 Hz, 107.79 Hz, 215.58 Hz, 431.16 Hz, 862.32 Hz, 1638.40 Hz, 3276.80 Hz, 8192 Hz, and 16384 Hz as they apply to one dimensional speech signal processing. From this understanding of how Gabor transforms work, the research extends to two dimensional analysis of Gabor transforms of Forward Looking InfraRed (FLIR) images. Self similar versus non-self similar Gabor transforms at frequencies corresponding to 1, 2, 3, 4, and 5 cycles and orientations of 0, 45, 90, and 135 degrees at each frequency were investigated using a digital simulation algorithm developed for this research. The digital simulation portion of this research produced a complete segmentation system using Gabor "filters". A preliminary feasibility study of the Gabor coefficients classification system based on jets as described by Flaton and Toborg (5:I313-I320) was also conducted. The research also investigated the requirements needed to optically generate Gabor transforms for image processing.

Outline of Thesis. Chapter II discusses fundamental concepts involved with the Gabor function, Gabor's original papers, and reviews recent work in the area of image segmentation and classification based on Gabor functions. The AFIT-developed Gabor based FLIR segmenter/classifier is described in Chapter III with discussion of results in Chapter IV. A conclusion and recommendations for further research are included in Chapter V.

II. Background

Summary of Fundamental Concepts and Related Research.

Mathematical Justification of the Gabor Function. If the spatial variables x and y, and their respective spatial frequencies, α and β, are functionally independent, then the form of the two dimensional Gabor transform is the correlation integral whose cosines and sines have been modified by the Gaussian distribution

$$F(x,y,\alpha,\beta) = \iint f(x',y')\, g^{(x,y,\alpha,\beta)}(x',y')\, dx'dy' \quad (1a)$$

where

$$g^{(x,y,\alpha,\beta)}(x',y') = e^{i\alpha x/2\sigma} e^{i\beta y/2\sigma} e^{-i\alpha x'/2\sigma} e^{-i\beta y'/2\sigma} g(x'-x)g(y'-y) \quad (1b)$$

and

$$g(x'-x) = \text{NORM}\ \exp(-(x'-x)^2/2\sigma^2) \quad \text{NORM = normalization constant}$$

$$g(y'-y) = \text{NORM}\ \exp(-(y'-y)^2/2\sigma^2). \quad (1c)$$

The function f(x',y') is the original two dimensional input image. The function $F(x,y,\alpha,\beta)$ is the two dimensional output image, the Gabor transform of the input image. The Gabor function, $g^{(x,y,\alpha,\beta)}(x',y')$, is the two dimensional, modified complex sinusoid

whose frequency variables, α and β, are related to the standard deviation of the Gaussian functions.

The relationship given in equation 1 displays all possible shifts for a given x' and y' hence the notation (x,y,α,β) for the correlation integral. The Gabor transform represents the correlation between the Gabor function and a particular two dimensional signal. The correlation between the input signal and the particular Gabor function of interest can be obtained by multiplying the Fourier transform of the image with the Fourier transform (complex conjugate of the Fourier transform) of the Gabor function. Recall,

$$f * g \quad \langle\!\!-\!\!\rangle \quad F \cdot G \tag{2a}$$

$$f \underline{*} g \quad \langle\!\!-\!\!\rangle \quad F \cdot G^{*} \tag{2b}$$

where * denotes convolution, $\underline{*}$ denotes correlation and * denotes complex conjugate (8:1-287).

Therefore, theoretically, Fourier transforming the input signal and the Gabor function, multiplying the Fourier transform of the input signal with the complex conjugate of Fourier transform of the Gabor function, then inverse Fourier transforming the product will result in the correlation between the particular Gabor function

of interest and the original space domain representation of the input signal. Realize that the output image from the inverse Fourier transform will be flipped about both the x and y axes relative to the input image.

A more generalized mathematical discussion of the class of functions to which the Gabor function belongs can be found in a paper by Grossmann and Morlet (10:723-736).

Grossmann and Morlet's discussion of square integrable functions, which includes the Gabor function and its functional form, namely its Fourier transform analog, presents the generalized mathematical foundation for using the functional form of the Gabor function.

The reader should be aware of the notational changes and that the arguments presented are derived for one dimensional Gabor transforms.

According to Grossmann and Morlet, an arbitrary complex-valued square integrable function s(t) can be represented by Gaussians, shifted in both the spatial and spatial frequency domains. If $g(t)=2^{-1/2}\pi^{-1/2}e^{-t^{*}t/2}$ and t', Ω' are arbitrary real variables, then

$$g^{(t',\Omega')}(t)=e^{i\Omega' t'/2}e^{-i\Omega' t/2}g(t-t') \qquad (3)$$

where $g(t-t') = 2^{-1/2}\pi^{-1/2}\exp(-(t-t')^2/2)$ is a normalized, one dimensional Gaussian which forms the inner product (dot product) between the square integrable function and the input

$$S(t',\Omega') = \int_{-\infty}^{+\infty} s(t)g^{(t',\Omega')}(t)\,dt \qquad (4)$$

By definition, the integral over all space for the magnitude of the given function, $|g|$, is unity, hence

$$\iint_{R^2} |S(t',\Omega')|^2 dt'd\Omega' = \int_{R} |S(t)|^2 dt \qquad (5)$$

In other words, the energy within the Gabor transform is that which is derived from the energy within the image.

The function $s(t)$ can be recovered from the function $S(t',\Omega')$ through

$$s(t) = \iint_{R^2} S(t',\Omega')g^{(t',\Omega')}(t)\,dt'd\Omega' \qquad (6)$$

which represents the inverse Gabor transform.

Through close inspection, the form of the square integrable transform as described by Grossmann and Morlet and that found in equation 1 produce the same mathematical results

$$F(x,y,\alpha,\beta) = \iint_{R^2} f(x',y')\, g^{(x,y,\alpha,\beta)}(x',y')\, dx'dy' \quad (1)$$

$$S(t',\Omega') = \int_{R^2} s(t) g^{(t',\Omega')}(t)\, dt \quad (4)$$

Letting t' equal x, Ω' equal α, S equal F, and extending the relationship in equation 4 to two dimensions by introducing y' and β, equation 4 becomes identical to equation 1.

Hence, the form of the Gabor transform is the correlation integral which is a member of an entire class of functions, the square integrable functions, producing transforms that may be useful for image processing.

Gabor's Original Work. The basic concept of multivariate frequency analysis was developed by Dennis Gabor in 1946. During the postwar decade, the ability to compress large amounts of information into fixed frequency bands was crucial for the

advancement of communications. In his three part paper, Gabor discussed how to compress and expand information to transmit over communication channels in an effort to optimize frequency band use.

Based on available experimental data, Gabor noted that human hearing loses its efficiency to discriminate frequencies beyond 1000 Hz, "... proving that sound reproductions which are far from faithful may be perceived by the ear as perfect, and that condensed methods of transmission and reproduction with improved waveband economy are possible in principle."

Gabor developed two means of compressing data for transmission over communications channels. Both approaches used a Gaussian envelope possessing a sinusoidal carrier whose frequency of variation was a direct function of the Gaussian standard deviation. By functionally linking the frequency to the Gaussian standard deviation, Gabor was able to produce a single function that possessed highly specific, tunable frequencies for speech compression and expansion [This type of Gabor function is commonly called self similar, which possess a fixed number of periods within the Gaussian envelope] (6:429-457).

M. R. Turner's work with modeling the human visual system. From Gabor's work with multivariate frequency analysis, M. R. Turner developed a model for the human visual system. Turner described the

problem of modeling the visual system as one of information representation. Both the single valued pixel representation and the Fourier transform representation provide complete image information. However, the pixel representation did not emphasize information about the patterns contained within the set of pixels, such as periodicity and structural distribution. The Fourier transform representation did not emphasize local gradients (distributions). Hence, the Fourier transform was not an efficient means of local textural discrimination since it could not faithfully reproduce fine structural detail, localized gradients.

The key, according to Turner, was to develop a method to represent the image completely in space and in frequency. Turner employed the Gabor function to provide a means to accomplish this task. Instead of using self similar Gabor functions, where the sinusoidal variation was directly proportional to the standard deviation of the Gaussian envelope, Turner used a constant Gaussian distribution and varied the sinusoidal frequency, the non-self similar Gabor function. By employing a non-self similar Gabor function, Turner was able to optimally display detailed textural features without sacrificing global features such as periodicity and overall data structure (23:71-82).

Daugman's work using Gabor functions. In 1988, John Daugman developed a neural network implementation of Gabor transforms for data reduction and image segmentation. As in Turner's work, Daugman expressed the need to explicitly represent features required to completely characterize a signal.

Daugman discussed three levels of image processing and related them to orders of correlation. Processing locally similar pixels corresponded to first order correlation, or low level correlation. Processing spatially continuous edges corresponded to second order correlation, or a level of correlation producing localized structural information. Processing uniformly distributed frequency corresponded to high order correlation, or global structural information processing. In other words, the level of correlation increases as the processing of the image becomes more and more global.

Daugman used the Gabor transform function because it provided the vehicle for extracting local spectral information (texture and form) while retaining information about the global "patterns" or textures of the image. Localization of information is accomplished through the "windowing" effect caused by the Gaussian envelope. Global textural information retrieval is accomplished using specific sinusoidal frequencies and orientations which "match" the global textural distribution.

Daugman described two methods for generating Gabor functions. The first method generated a constant Gaussian standard deviation with a varying sinusoid carrier. This type of Gabor function, based on Bastiaan's use of biorthogonal functions (1:538-539), and known as a non-self similar Gabor function, differs from the self similar Gabor function as described by Gabor in that the sinusoidal variation is not a function of the standard deviation of the Gaussian envelope (see Figure 1.1 and 1.2 for examples of non-self similar and self similar Gabor functions). The non-self similar method allowed Daugman to "search" the entire image for local texture differences to find regions of highest probability for object location. The second method, the self similar Gabor transform method, allowed Daugman to concentrate computational effort on specific regions of the image to better define the textural content of that region (4:1169-1179).

Mallat's work with "Gabor-like" functions. Mallat studied the properties of a set of wavelet functions belonging to the $L_2(R^n,R)$ class of functions. The $L_2(R^n,R)$ class includes all square integrable functions such as the Gabor function as well as Mallat's orthonormal basis set of wavelets.

The decomposition of Mallat's wavelets using the square integrable function set defines the orthogonal multiresolution

representation (wavelet representation). The important feature of Mallat's wavelet representation is its orthogonality since Gabor transforms do not constitute a complete, orthogonal set of basis functions. Orthogonality implies non-redundancy in the transform output and also affords the ability to formally derive a sampling theorem based on the orthogonal set of functions.

Mallat's pyramidal algorithm, which is based primarily on the Fourier convolution integral, differentiates several spatial orientations at various levels of resolution. For two dimensional imaging, Mallat used a well known technique called "turning the corner". He exploited the fact that the convolution kernel was separable in the two dimensions, x and y. First, Mallat convolved the rows of the image with the one dimensional filter, retained the values of the convolution, then performed the same operation using the columns of the retained row convolution values. Mallat described these filters (convolution operations) as quadrature filters or conjugate mirror filter decomposition.

Using this technique, Mallat was able to show orientation specific, texture discrimination for well defined digitized images of boxes on top of tables, toy model buildings, and computer generated preattentive images (images possessing geometric patterns that can be detected with little conscious human effort).

Mallat's work concentrated on data compression for image processing which he showed that with less than 1.5 bits per pixel, the pristine images could be reconstructed with few visible distortions. Furthermore, with *a priori* knowledge of the signal's statistical intensity distribution, Mallat proposed that optimization of the code for signal processing could be achieved (16:674-693).

Flaton and Toborg's Sparse Filter Graphs for Image Recognition. Flaton and Toborg developed a segmentation and recognition scheme based on two dimensional Gabor functions. The recognition of targets within non-simulated FLIR images is based on the construction of feature vectors extracted from correlations of the image with the Gabor function at various orientations and spatial frequencies.

Flaton and Toborg use of self similar Gabor functions was primarily biologically motivated. There is evidence which suggests that Gabor functions, or "Gabor-like" functions exist in the visual cortex of cats (12:1233-1258). The self similar Gabor functions they used consisted of four frequencies and six evenly spaced orientation ranging over 180 degrees for a total of 24 Gabor functions from which the "jets" were generated.

Each set of coefficients generated by correlating the Gabor functions with the image was "stacked" in a hierarchical structure called "jets" (see Figure 2). Jets are vectors made from the coefficients generated by correlating the Gabor function of specific orientation and spatial frequency with the particular image of interest. The stacks were centered about a point in the original image. Therefore, the "jets" represented the localized Gabor correl tion over a neighborhood which possessed an object of interest.

If a jet were generated for every pixel in a 128 by 128 image, this set of data would be called a Gabor block. Representing an image of this size, with four frequencies and six orientations, would require 128 pixels by 128 pixels by 4 frequencies by 6 orientations, or nearly 400,000 elements. Therefore, Flaton and Toborg devised a method for determining which "jets" produced the "most information", then stored only those "jets" possessing this characteristic.

Flaton and Toborg measured the likeness of one jet to another in terms of direction and length. Similarity (likeness), combined with two selection rules, was used for determining the characteristic of "most information" (saliency). The proper choice of the saliency measure determined the effectiveness of subsequent matching procedures.

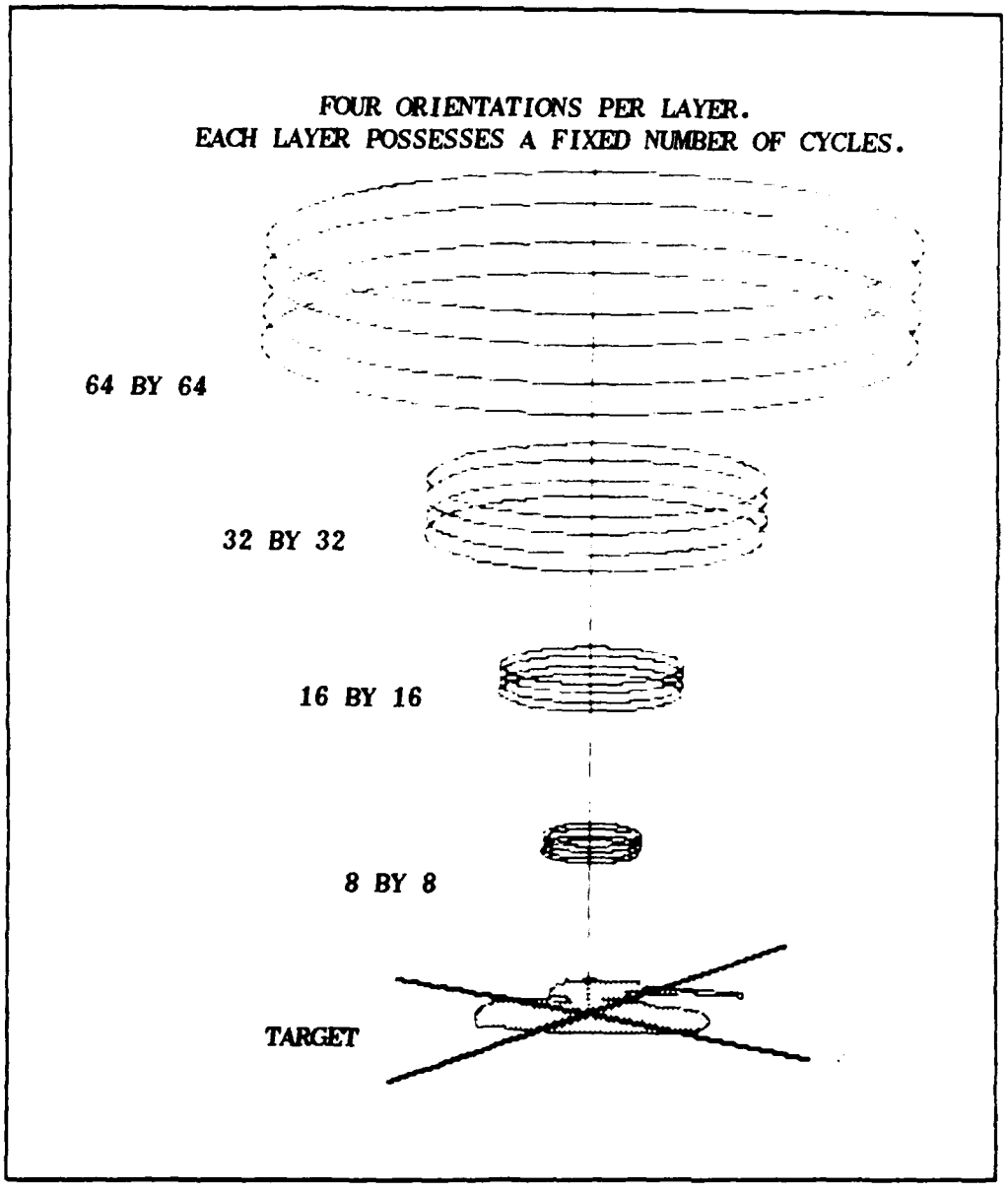


Figure 2. Jets. The hierarchical "stacking" of Gabor transform coefficients. Each layer corresponds to a spatial frequency. Each layer possesses a given number of spatial orientations.

The equation for determining saliency as defined by Platon and Toborg is

$$S(J^a,J^b) = \frac{J^a \cdot J^b}{(||J^a||\ ||J^b||)} * \min\left[\frac{||J^a||}{||J^b||}, \frac{||J^b||}{||J^a||}\right] * \min(||J^a||,||J^b||) \quad (7)$$

where J^a, J^b are two different jets.

The first term is the standard dot product of two vectors with enhancement if the jets have the same direction. The first selection rule (the first "min" term in equation 7) compares the reciprocal ratios of the magnitudes of the jets. If J^a is much larger than J^b, then the first selection rule will diminish the saliency value.

The second selection rule (the second "min" term in equation 7) will select the minimum value between the magnitudes of the jet vectors. Hence, if the magnitudes of J^a and J^b are large, then this selection rule will emphasize this characteristic. Therefore, if jets are roughly the same, then they will exhibit high saliency, and be subject to elimination.

By generating and storing only those jets possessing "highest information" content (Gabor mini-blocks), Platon and Toborg

demonstrated the potential for image recognition on 128x128 pixel, FLIR images of tanks. It should be noted that their research was limited to 100 FLIR images, each image containing a single target (M-60 or T-62 tank) which filled most of the 128x128 pixel image plane.

The recognition scheme used as its template one of the stored mini-blocks, to which other mini-blocks within the database were compared. Eight of the most "salient" jets within the Gabor mini-block of the template were iteratively chosen and compared to all jets within the mini-blocks of the test images. The matching was performed using the similarity (dot product) portion of the saliency measure. The test image with the highest set of matching jets was scored the correct response.

The Gabor transforms used by Platon and Toborg were not orthogonal, that is to say, the eigenvectors corresponding to the transformation from spatial domain to spatial frequency domain do not constitute an orthogonal set, hence some redundancy will be found in the transform output. Platon and Toborg could not determine the effect of the non-orthogonality of the Gabor transform on the processing of images. Because of the non-orthogonality of the Gabor transform, no derivation of a formal sampling theorem was possible, ie. no specific set of Gabor functions could reliably extract the information necessary to classify the targets. Only

through previous experimental results could Platon and Toborg determine which sets of frequencies and orientations of the Gabor functions would work on the FLIR images. [Note that Gabor transforms are not orthogonal and will lead to redundancies in the output] (5:I313-I320).

In summary, the mathematical background for the use of the functional form of the Gabor transform is provided by Grossmann and Morlet. The works of Gabor, Daugman, Turner, and others provide the foundation for using Gabor transforms for image processing. Platon and Toborg demonstrate a very limited application of Gabor transform coefficients to classify (recognize) targets. The following chapter will describe the methodology used to develop the Gabor transform segmenter and provide the initial investigation into using Platon and Toborg's method for classification.

III. Methodology

General Approach and Methodology.

The first preliminary goal of this research was to develop and implement a one dimensional Gabor filtering algorithm for speech analysis. This effort used MathCAD, a modeling simulation software package which imports and exports data, generates graphs, tables, and equations for implementing Gabor transforms (Appendix B). Appendix B shows a MathCAD template for Gabor transforming blocks of 512 data points corresponding to speech. This portion of the research culminated in producing the correlation between the speech signals and the Gabor function, the Gabor correlogram and yielded a better understanding of how the Gabor function worked.

The second preliminary goal of this research was to review, analyze, and assess a C-language code for generating self similar two dimensional Gabor transforms using the UNIX operating system. Concurrent to this effort was the designing, development, and implementation of a non-self similar Gabor transform for segmenting forward looking infraRed (FLIR) images using a C-language code. A separate effort was taken to develop code to run on the VMS operating system for FLIR image processing using FORTRAN (See Appendix C, parts 1 and 2, BSG.FOR and BCG.FOR, respectively).

The primary goal of this thesis consisted of testing and evaluating the computer simulation of the self similar and non-self similar Gabor transforms for their ability to segment FLIR images. The code developed for the VMS operating system was chosen for this research because of its ease of operation.

The tests were designed to examine the effects of varying parameters to enhance the Gabor's ability to segment raw FLIR images. These tests included varying the frequency and orientation of the Gabor transforms, varying the spatial extent of the Gabor function, investigating the effects of various in-plane rotations, image superposition, comparing sinusoidal variation versus cosinusoidal variation of the Gabor wavelets, then comparing the performance of self versus non-self similar Gabor transforms under these conditions. Once the test results were evaluated, the requirements for recognition based on parameters established during testing were developed.

Two Dimensional Gabor Transforms.

A block diagram of the Gabor based segmentation-classification system is shown in figure 3.1 and is also included as Appendix K.

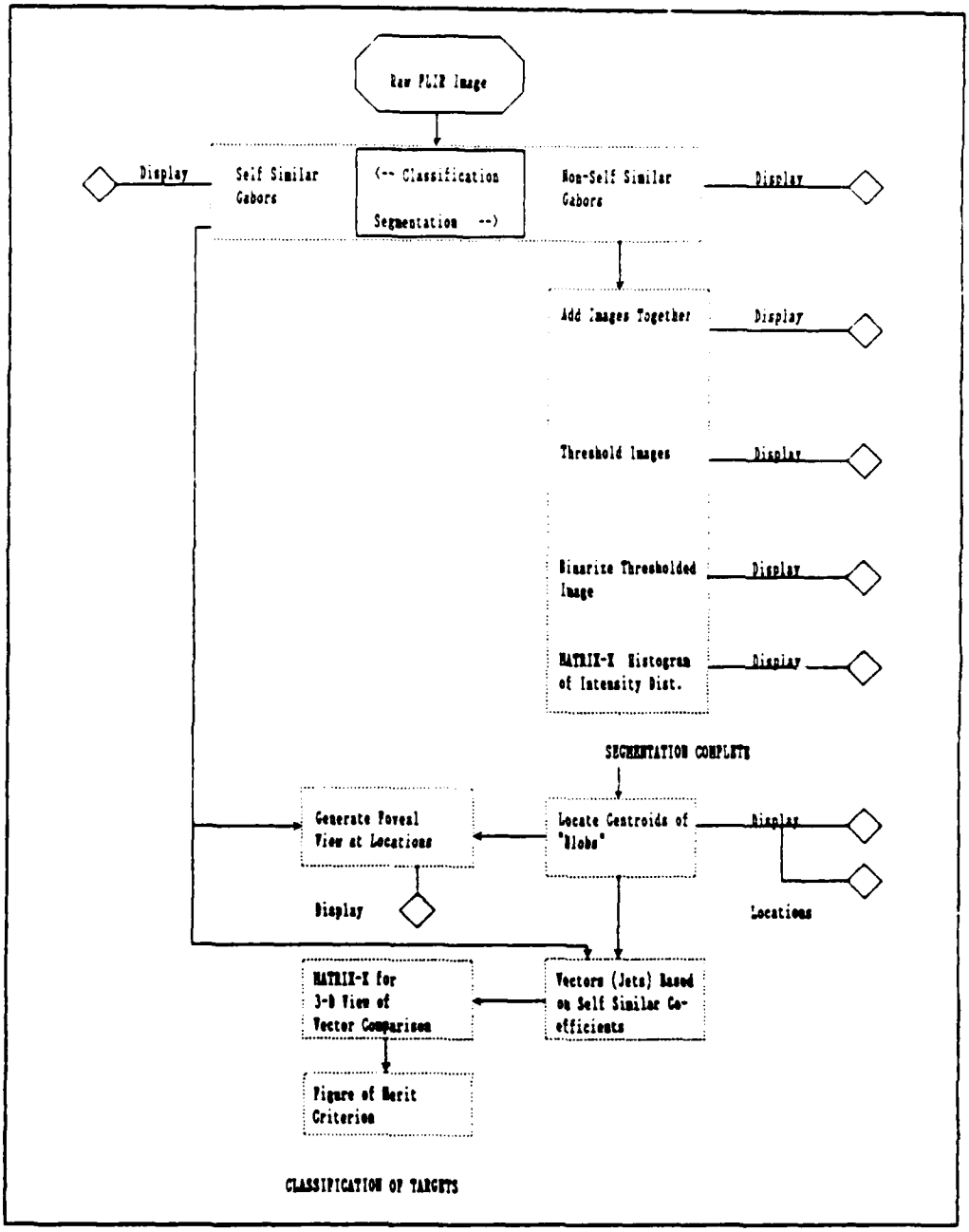


Figure 3.1 Gabor segmentation-classification system.

Two Dimensional Fast Fourier Transforms. The two dimensional Fast Fourier Transform (FFT) was the key to developing the two dimensional Gabor transform. The FFT routine used for two dimensional image processing was obtained from Gonzalez and Wintz (7:105-109).

The FORTRAN subroutine computed the FFT via a successive doubling technique whereby the input array was initially reordered (bit reversed for proper transforming) followed by the successive doubling calculations with the result divided by the length of the input vector.

As noted by Gonzalez and Wintz, since the FFT subroutine requires complex data format, any real input data vector had its quadrature term set to zero prior to being processed by the FORTRAN code.

The method of "turning the corner" was used to accomplish the 2-D FFT. Turning the corner involved performing a one dimensional FFT of the columns of the data array representing the image followed by performing a one dimensional FFT of the rows of the result of the first 1-D FFT. It should be noted that performing the FFT on the rows first followed by the FFT of the columns produces an identical result.

Since the 2-D Fast Fourier Transform is separable in its two dimensions, performing the second FFT operation on the result of the first FFT is mathematically correct.

Image input. The Forward Looking InfraRed (FLIR) images were stored in 240 by 640 pixel arrays as BYTE or LOGICAL*1 integers whose pixel values were accordingly restricted to a range of -128 to 127. The BYTE or LOGICAL*1 format allowed efficient storage of images. Each image data set was converted to INTEGER format before manipulation by the Gabor transform code.

Normalization of input image. Since energy normalization was a serious concern for proper image processing, the input image was normalized by computing the maximum pixel value within the given input image, dividing every pixel value by the maximum, then multiplying every pixel by 255 to produce a scaled representation of the input image.

PLIR images were also normalized using the Lambert technique (15:1-86). Results and discussion about the Lambertization technique are found in Appendix O.

Gabor Function Generation. All Gabor functions were generated "On Center" in a 256 by 512 array using the FORTRAN code found in Appendix C. Since both extents of the array are even, a center pixel does not exist. However, all Gabor functions were centered ± N/2 about the pseudo-center of the array.

Three types of Gabor functions were generated for image processing. The first type was the self similar Gabor function. Self similarity implies that the deviation of the Gaussian envelope directly influences the sinusoidal variation therefore maintaining a like number of periods under the curve (similar frequency distribution) irrespective of the spatial extent of the Gabor function.

The spatial extent, N, of the Gabor function (pixel by pixel extent) was a parameter interactively given to the computer code during execution of the Gabor transform FORTRAN program. The spatial extent determined the standard deviation of the Gaussian cofactor and, consequently, the frequency of the sinusoidally varying cofactor.

The second type of Gabor function used was the non-self similar Gabor function. The standard deviation of the Gabor function's Gaussian cofactor is not related to the frequency of the sinusoidal cofactor therefore, with a constant Gaussian envelope maintained, a varying number of periods under the curve (frequency distribution of the Gabor function) can be achieved. A criterion of four standard deviation units at the ± N/2 points was used to ensure zero (or near zero) pixel values at the maximum spatial extent of the Gabor functions.

The third type of Gabor function generated was the modified, or multi-purpose Gabor function. With the multi-purpose Gabor function, the standard deviation of the Gaussian cofactor is a scaled equivalent to the sinusoidal cofactor's frequency, thereby combining the dilating characteristic of self similar Gabor functions with the multiple frequency characteristic of non-self similar Gabor functions.

Sine wave Gabor transforms were generated using a batch mode submitted FORTRAN program called BSG.FOR (Appendix C, part 1). Cosine wave Gabor transforms were generated using a batch mode submitted FORTRAN program called BCG.FOR (Appendix C, part 2).

Image Output and Storage. The output of the Gabor transform FORTRAN code was the intensity or the Gabor filtered image, where each pixel value corresponded to a percentage of the maximum intensity. All output images were converted to BYTE format for two-dimensional array storage by employing the MOD conversion to values between -128 and 127.

Multiple Image Superposition.

FORTRAN code was developed to add together (superimpose) Gabor filtered images (Appendix D, THRESH_ADD_HIST.FOR). Histograms of the intensity distribution within the superimposed images determined

a threshold level for the filtered images. THRESH_ADD_HIST code added, pixel for pixel, multiple images, the exact number and names of images determined by the user. The superimposed images were histogrammed to find a thresholding level suitable for binarization.

Table 1. Bin number relative to pixel intensity value.

BIN NUMBERS			ASSOCIATED PIXEL VALUE RANGES		
1	20	39	0	90-94	185-189
2	21	40	1-4	95-99	190-194
3	22	41	5-9	100-104	195-199
4	23	42	10-14	105-109	200-204
5	24	43	15-19	110-114	205-209
6	25	44	20-24	115-119	210-214
7	26	45	25-29	120-124	215-219
8	27	46	30-34	125-129	220-224
9	28	47	35-39	130-134	225-229
10	29	48	40-44	135-139	230-234
11	30	49	45-49	140-144	235-239
12	31	50	50-54	145-149	240-244
13	32	51	55-59	150-154	245-249
14	33	52	60-64	155-159	250-254
15	34	53	65-69	160-164	255
16	35		70-74	165-169	
17	36		75-79	170-174	
18	37		80-84	175-179	
19	38		85-89	180-184	

The histogramming algorithm separated each pixel value into bins according to its value. Bin one corresponded to all pixel

values of zero. Bin two corresponded to values ranging from one to four. Bin three corresponded to pixel values ranging from 5 to 9, bin four corresponded to pixel values ranging from 10 to 14, and so on until bin 53 which corresponded to the value 255. Table 1 shows the relationship of bin number to pixel range.

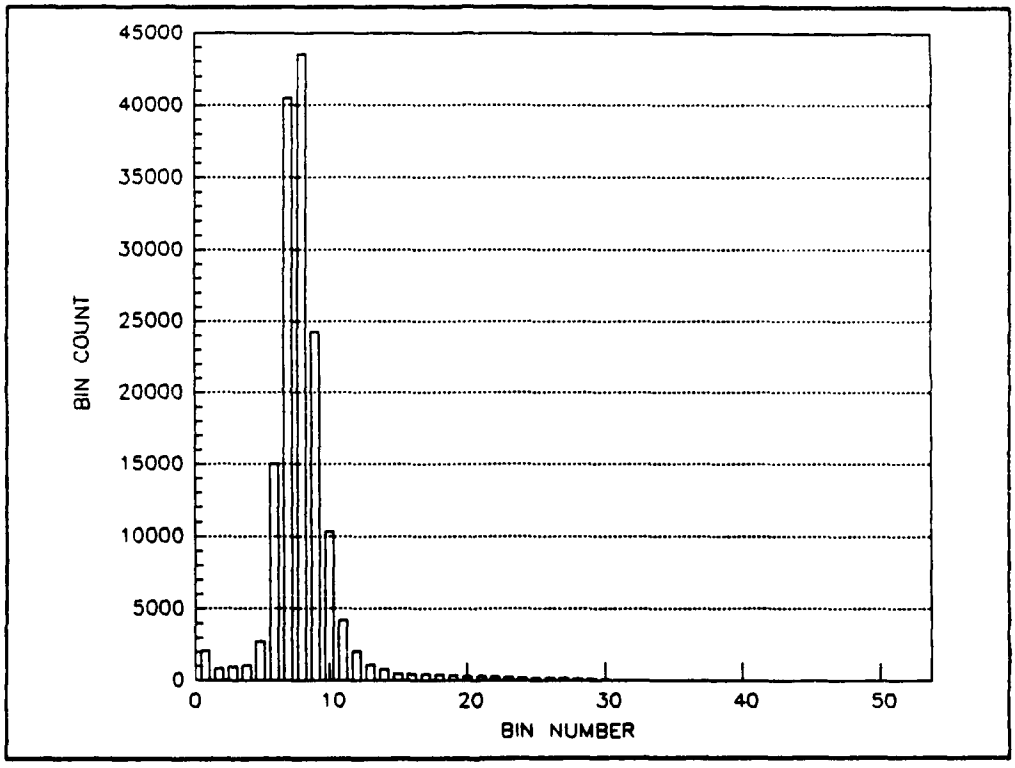


Figure 3.2. Histogram of the intensity distribution for FLIR image REFK41.FLR. Distribution of intensity is nearly Gaussian.

Figure 3.2 is an example of the histogram of the intensity distribution for FLIR images (in particular, REFK41.FLR). Figure

3.3 is an example of the intensity distribution for a Gabor filtered FLIR image (in particular, FLIR image: REFK41.FLR. Gabor filter: frequency 3, orientations 0, 45, 90, and 135 degrees).

The bin containing the most number of pixels determined the initial lower bound on the threshold. For the distribution found in figure 3.3, the initial lower bound was bin number 7 (pixel intensity values ranging from 24 to 29).

Recall, each bin corresponds to a certain intensity level. Hence, the bin with the most pixels corresponded to the lower bound on the intensity threshold. Each bin above the maximum count bin was compared to a count level set by the user.

For instance, if the user desired all bins having number of pixel counts greater than 1000 to be discarded, then the user would input 1000 as a count level. Using the bin count as a threshold for image segmentation assumes that, if the segmentation algorithm is correctly working, the number of pixels associated with the target should be significantly less than those associated with the background.

Once the bin count dropped below the count level set by the user, the upper bound on the threshold was set at this bin location. For the example in figure 3.3, the upper bound was bin number 10 (pixel values ranging from 40 to 44). The algorithm then searched all bins prior to the bin with the most number of pixels. The first

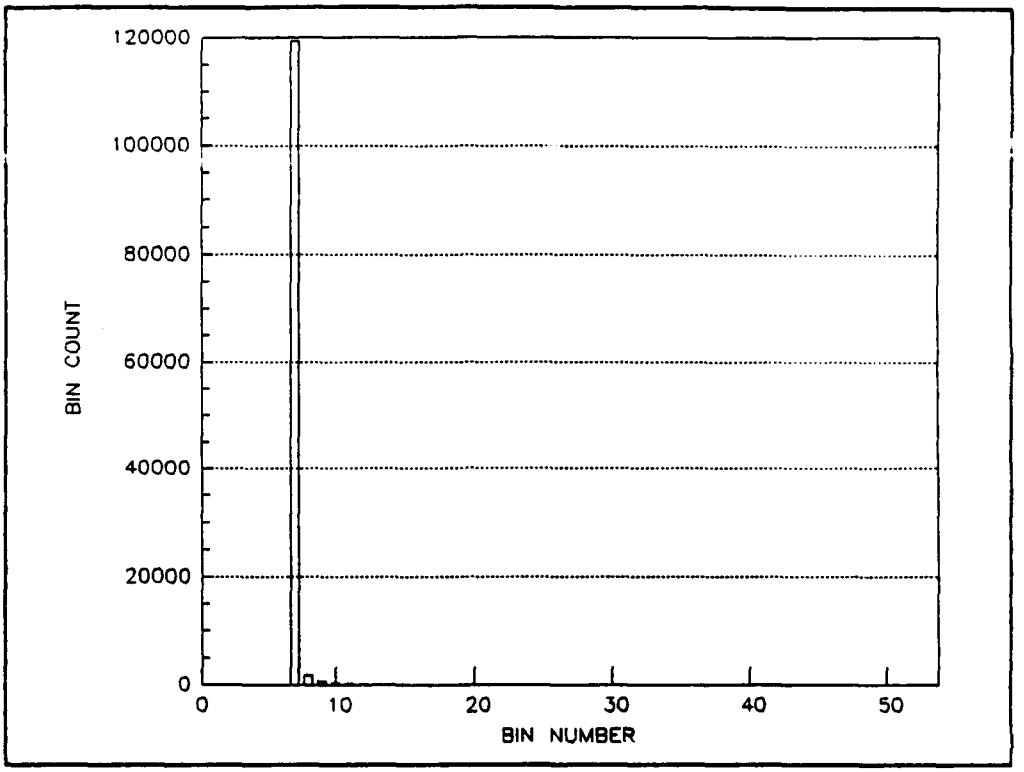


Figure 3.3. Histogram of intensity distribution for Gabor filtered image (FLIR image: REFK41.FLR, Gabor: freq: 3, orient: 0, 45, 90, and 135 degrees). Distribution is centered about the global average intensity, facilitating the segmentation process.

bin with more than 1000 pixels was determined to be the lower limit to the threshold. If no bin met this criterion, then the bin with the most number of pixels was determined to be the lower bound to the threshold. For the example in Figure 3.3, the initial lower bound was the final lower bound. Note the scale difference between Figure 3.2 and Figure 3.3. In Figure 3.3, most of the pixels reside

in one bin allowing easy thresholding. The distribution of pixels over the intensity spectrum is more Gaussian in Figure 3.2, thus giving rise to the difficulty in choosing a proper threshold level.

Once the lower and upper threshold levels were found, the resulting superimposed images were thresholded using a pass/no-pass filter. All pixels in bins above the upper threshold and all pixels below the lower threshold were set to unity. Pixels in the other bins were set to zero.

THRESH_ADD_HIST code generated MATRIX-X readable files for plots of the histograms on screen using a FORTRAN subroutine provided by Integrated Systems, Inc., makers of MATRIX-X. MATRIX-X is an application software package specifically tailored for linear algebra.

Labeling regions of interest.

The binarized output of the THRESH_ADD_HIST program was fed into a FORTRAN program to find the center pixel of regions of interest (APPENDIX B, SHOW_LAB_FLIR_IM.FOR). SHOW_LAB_FLIR_IM found the maximum extent of any region of interest in both directions, then computed the center pixel of each of these regions.

After finding the regions of interest, SHOW_LAB_FLIR_IM designated all blobs within the regions as a certain number. For

instance, the blob corresponding to region one was labeled as one, the blob in region two as two, and so on.

The output of this program was a listing of regions of interest with locations of the center pixels of each of these regions. In addition to the listing, a display of the regions of interest was also provided by this program.

Computing jets centered on regions of interest.

The locations of regions of interest were fed into a FORTRAN program that computed the "jets" centered on these locations (Appendix G, JETS.FOR). Recall a jet is a set of self similar Gabor functions of varying sizes (spatial extent, hence spatial frequencies) and orientations within the varying sizes superimposed over a center point (regions of interest provided by the SHOW_LAB_FLIR_IM program).

For this research, four sizes ranging from 128x128 to 16x16 in octaves and four orientations ranging from 0 to 135 degrees in 45 degree increments were used such that, for any given jet, there were 16 layers per jet.

Since the Gabor transform code (Appendix C) already generated the self similar Gabor filtered images, the construction of the jets was reduced to merely windowing about the center points found by the

SHOW_LAB_FLIR_IM program and superimposing the windowed portions of the self similar Gabor filtered images.

Recognition of targets based on jets.

Twenty Dimensional Jet Vector Approach. Each jet consisted of twenty components corresponding to the center pixel values of correlation between the various frequency and orientations of the self similar Gabor filters and the raw FLIR image, center pixels generated by the SHOW_LAB_FLIR_IM program. The jet vectors were normalized by computing the difference between a given value and the minimum value then dividing by the maximum extent of the component values

$$\text{New Component Value} = \frac{\text{Original Component Value} - \text{Minimum Value}}{\text{Maximum Value} - \text{Minimum Value}} \quad (8)$$

thus setting the maximum valued component to unity and the minimum valued component to zero. The FORTRAN program to generate the vectors is found in Appendix H (VECT_JETS.FOR).

Twenty-dimensional vectors used for exemplar set one were taken from the jets of image REFM13.FLR for each target type (tank, truck, APC, and POL) present within the image. REFM13.FLR did not

possess POL vehicles. The jets obtained from REFM13.FLR were "concatenated" into a matrix format readable by MATRIX-X and compared to "concatenated" jets of images possessing targets of similar size but differing in aspect angle and position within the image.

A simple Minkowski two space, or Euclidean, distance rule was used as baseline a comparison measure between vectors (see Appendix I). Euclidean distance is the square root of the sum of the square of the difference between vectors

$$\text{Eucl. Dist.} = \left(\sum_{i=1}^{n} (\text{exemplar}_i - \text{test}_i)^2\right)^{\frac{1}{2}} \qquad (9)$$

where exemplar_i is the ith component of the exemplar vector and test_i is the ith component of the test vector. The difference producing a minimum distance was deemed the correct response.

This approach used vectors from REFM13.FLR for the exemplar set and compared this set to vector sets from REFM14.FLR, REFM15.FLR, REFM16.FLR, REFM17.FLR, REFM18.FLR, from REFM19.FLR.

A figure of merit (F.O.M.) was assigned to the distances between the exemplar vector(s) and the test vector(s). The figure of merit was calculated using the following rule:

$$F.\ O.\ M. = \frac{\text{Minimum distance (x)}}{\text{Minimum distance (1)}} \qquad (10)$$

where x ranges from 2 to 6. The larger the figure of merit value, the more likely the minimum distance (1) is the correct choice.

Eight Dimensional Jet Vector Approach. This approach used vectors from REFM14.PLR for the exemplar set and compared this set to vector sets from those used in the twenty dimensional jet vector approach. This approach put limitations on the twenty dimensional jet vector approach. First, regional locations found using the Roggemann centroid routine were used to determine specific locations from which the vector (eight dimensional) comparison sets were constructed. The specific locations used were 1) Mid hull, side viewed tank, foreground, 2) Rear cog, side viewed tank, foreground, 3) Front cog, side viewed tank, foreground, 4) Left tread, front viewed tank, foreground, and 5) Right tread, front viewed tank, foreground.

Second, the Roggemann centroid finding algorithm (SHOW_PLIR_IM) found the centroid of regions corresponding to possible target locations. The actual locations used for this approach were manually positioned to eliminate the "50% on - 50% off" problem encountered when using the twenty dimensional jet vector approach since the target encompassed most of the largest

window size (16 by 16) at these locations. Care was taken to ensure that the specific locations stayed relatively close to the original positions determined by the Roggemann centroid finding algoritnm.

Kth-Nearest Neighbor Approach. The kth-nearest neighbor approach involved the use of a C-language based code which determined the distances between vectors for first, third, fifth, seventh, and ninth nearest neighbors. The code randomly picked a vector and computed the distance to the first nearest neighbor, third nearest neighbor, and so on to the ninth nearest neighbor.

The program then calculated a figure of merit based on the ratio of distance calculated to minimum distance. The nearest neighbors were identified as to their respective vector designation and class (ie. vector 80, class 0, compared to the test vector designation of 0, class 0). The classification and accuracy of network choice were then displayed as a numeric designation and "right" or wrong" choice, respectively.

IV. Results and Discussion

The following chapter presents the results, with their respective discussions, from using the methods described in chapter III. The presentation is in three parts. The first part discusses the results of the segmentation portion of this research. The second part discusses the results of the preliminary investigation into classification based on Gabor jet coefficients. The third part proposes a possible optical architecture for accomplishing Gabor transforms.

Part 1. SEGMENTATION.

General Discussion. A general discussion of the results is presented to aid the reader in understanding overall effects of Gabor transforms on FLIR imagery.

Non-Self Similar versus Self Similar. Figure 4.1 shows the original FLIR image REFM13.FLR. From left to right, in the foreground, the targets in this image are front viewed tank, side viewed tank, and truck. In the background, the targets are, from left to right, armored personnel carrier and truck.

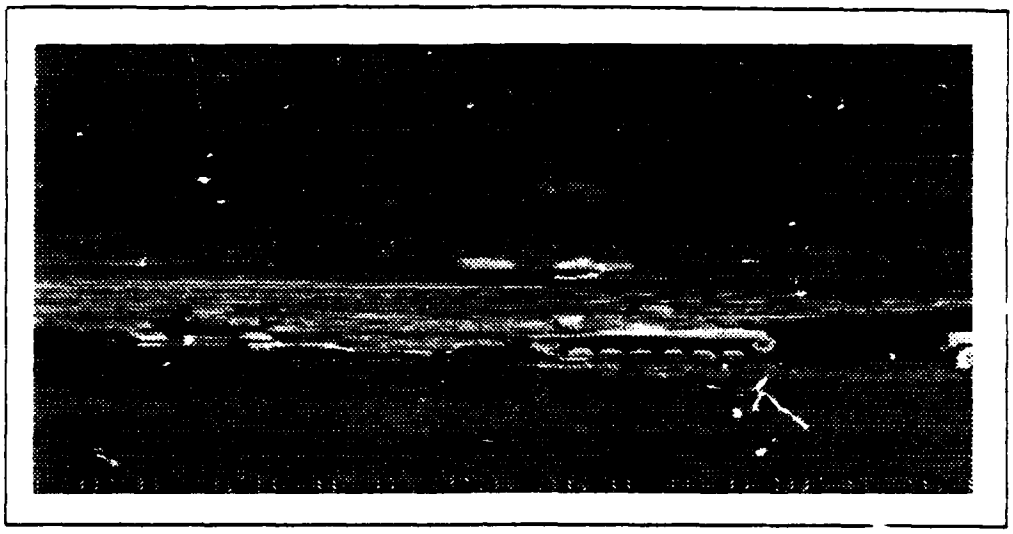


Figure 4.1. Original FLIR image, REFM13.FLR. Targets are ... background: APC and truck, foreground: front viewed tank, side viewed tank, and truck.

Figures 4.2 and 4.3 show the effects of filtering a raw FLIR image through self similar and non-self similar Gabor filters, respectively. Figure 4.2 is the superposition of the correlation between the raw FLIR image (REFM13.FLR) and self similar Gabor functions at frequency 3 and orientations of 0, 45, 90, and 135 degrees for a Gabor size of 16 by 16.

Figure 4.3 is the superposition of the correlation between the raw FLIR image (REFM13.FLR) and non-self similar Gabor functions at frequency 3 and orientations of 0, 45, 90, and 135 degrees for a Gabor size of 16 by 16.

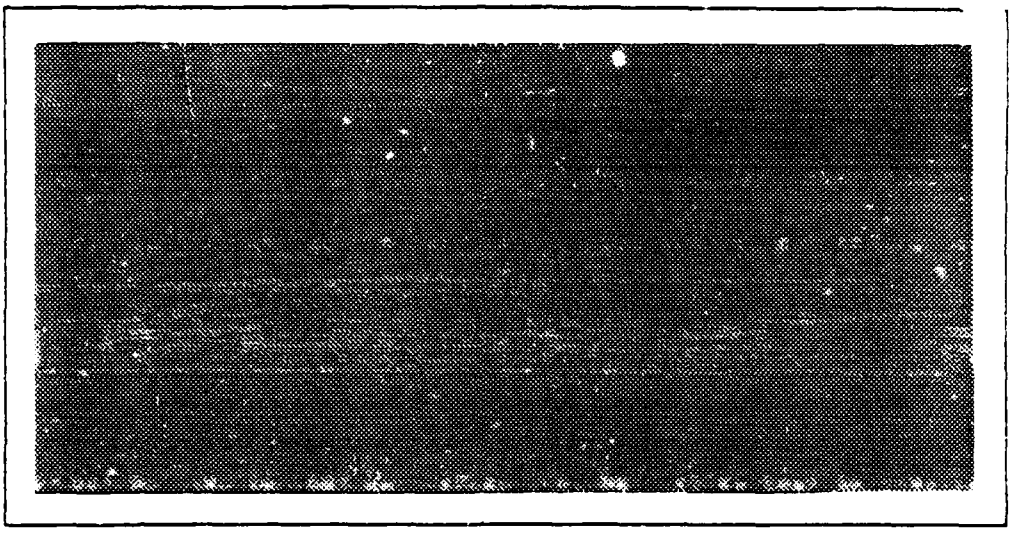


Figure 4.2. Correlations with self similar Gabor functions. Apparently, self similar Gabor functions cannot segment images. A caveat to this statement is found in Appendix K.

Apparently, as is seen in Figure 4.2, self similar Gabor functions do not possess the ability to segment the FLIR images used for this research effort (a caveat to this statement is found in Appendix M). This result is not unexpected since the spread of the Gaussian cofactor must be such as to produce a sinusoidal variation that "matches" the frequency content of the image while producing a "roll off' at the maximum extents to maintain proper localization of energy.

On the other hand, non-self similar Gabor functions segment the FLIR images quite well, as is seen in Figure 4.3. Processing

FLIR image REFM13.FLR at frequency 3/16 cycles per pixel and orientations of 0, 45, 90, and 135 degrees, REFM13.FLR image is

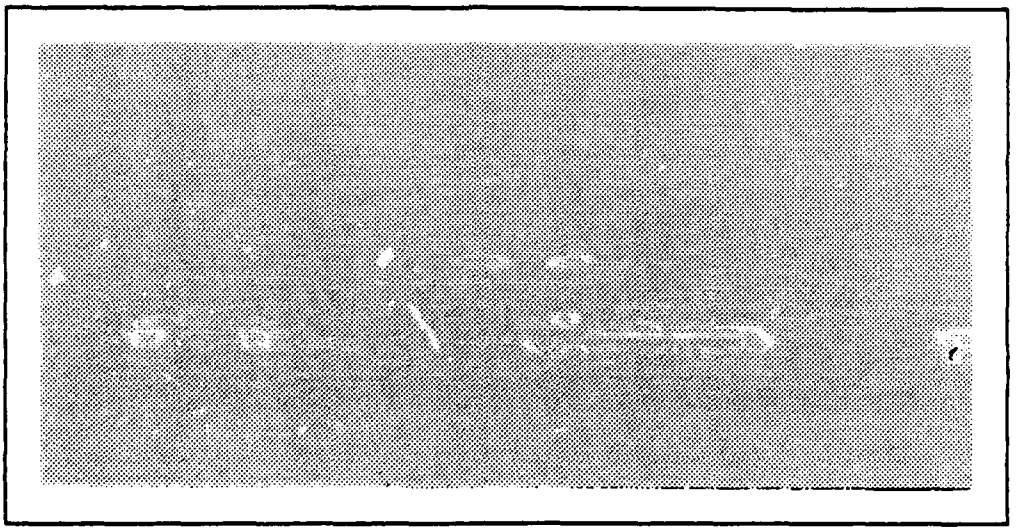


Figure 4.3 Correlations with non-self similar Gabor functions. Indications around the target are relatively high compared to those in Figure 4.2. This is due, in part, to ease of localization of energy.

reduced to only those regions possessing targets of interest. This result is to be expected since four standard deviations correspond to the maximum extent of the Gabor "patch" when using the non-self similar Gabor transform algorithm. Localization of the energy to within a small region is accomplished by limiting the spread of the Gaussian cofactor. Therefore, any contribution to the correlation from outside the Gabor "patch" is negligible.

Cosine wave variation versus Sine wave variation. For purposes of illustration, non-self similar Gabor functions were chosen for this section. Figure 4.4a shows the original FLIR image REFJ15.FLR. The targets in this FLIR image are, from left to right, a tank, an armored personnel carrier, a target board, and a truck. Figure 4.4b represents the binarized, non-self similar cosine wave Gabor filtering of raw FLIR image REFJ15.FLR at three cycles per envelope, orientations of 0, 45, 90, and 135 degrees superimposed. As evident in Figure 4.4b, the cosine wave Gabor function acts as a "body filler", paying more attention to the slower changing regions of the image than the edges.

Note, though, if the frequency of the cosine is high relative to the standard deviation (or spread) of the Gaussian envelope, then the cosine wave Gabor function acts very similar to the sine wave Gabor function yet retains its body filling capacity. However, when the frequency of the cosine approaches the "spread" of the Gaussian envelope, relative insensitivity to intensity is no longer a feature of the cosine wave Gabor function.

At first glance, this gain of dependence on the intensity present within a given image may seem detrimental. However, the other key feature to Gabor transforms, namely the frequency and orientation dependence, becomes extremely important.

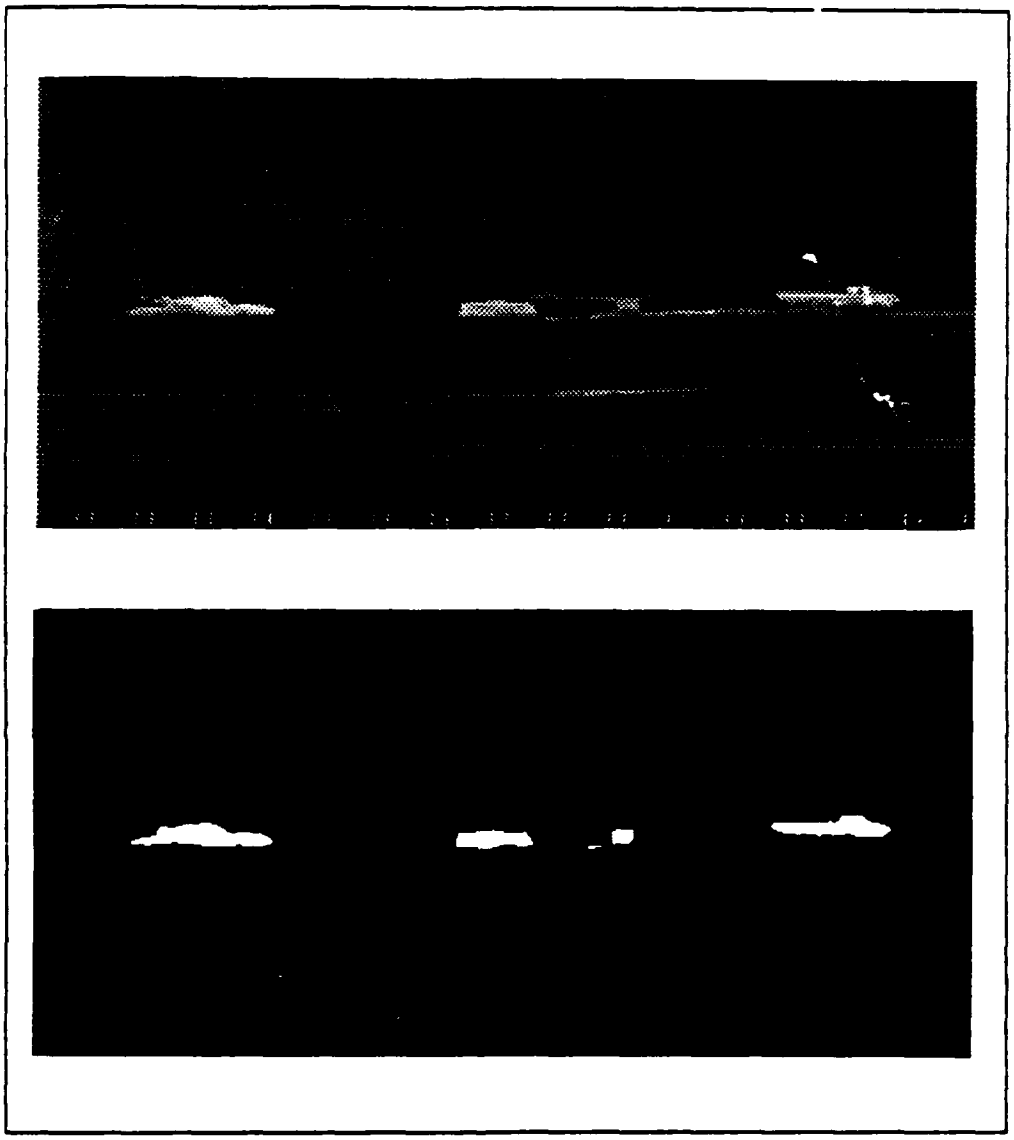


Figure 4.4. (a) Original FLIR image, REFJ15.FLR. Targets are, from left to right: tank, APC, target board, and truck (b) Binarized version of non-self similar cosine Gabor transform of FLIR image REFJ15.FLR.

Figure 4.5a is the superposition of raw FLIR image REFM13.FLR and a test image comprised of non-self similar, 3/16 cycles per pixel cosine Gabor functions (orientations 0, 45, 90, and 135 degrees) and two blocks of high intensity pixel roughly located 1/4 the distance from the top of the image and 1/4 from the bottom of the image.

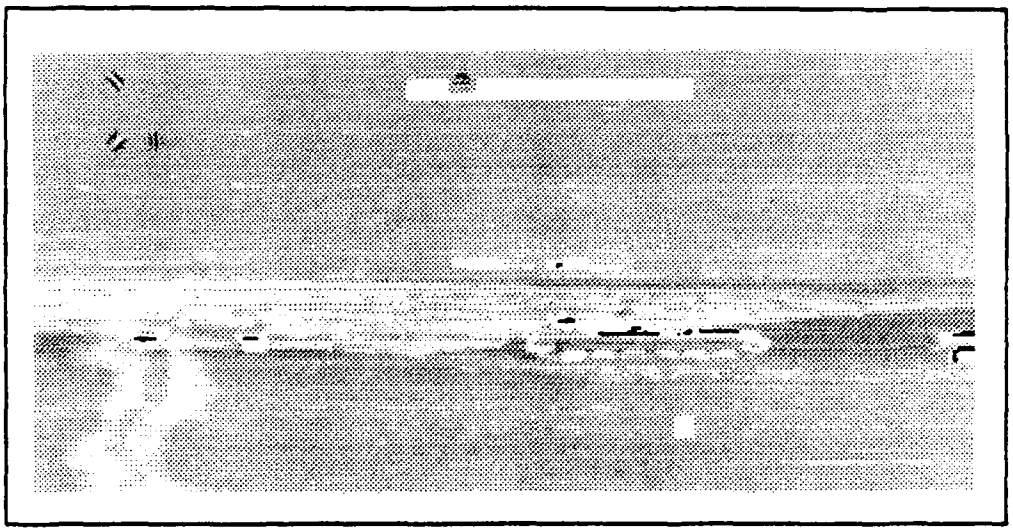


Figure 4.5a. REFM13.FLR with artificially added high intensity blocks and 0, 45, 90, and 135 degree, 3 cycle cosine Gabor functions.

By choosing the proper cosine frequency and orientations (frequency 3/16 per pixel and orientations 45 and 135), the other structures are passed the high intensity blocks, the tank, truck, and other targets within the image preserved (see Figure 4.5b).

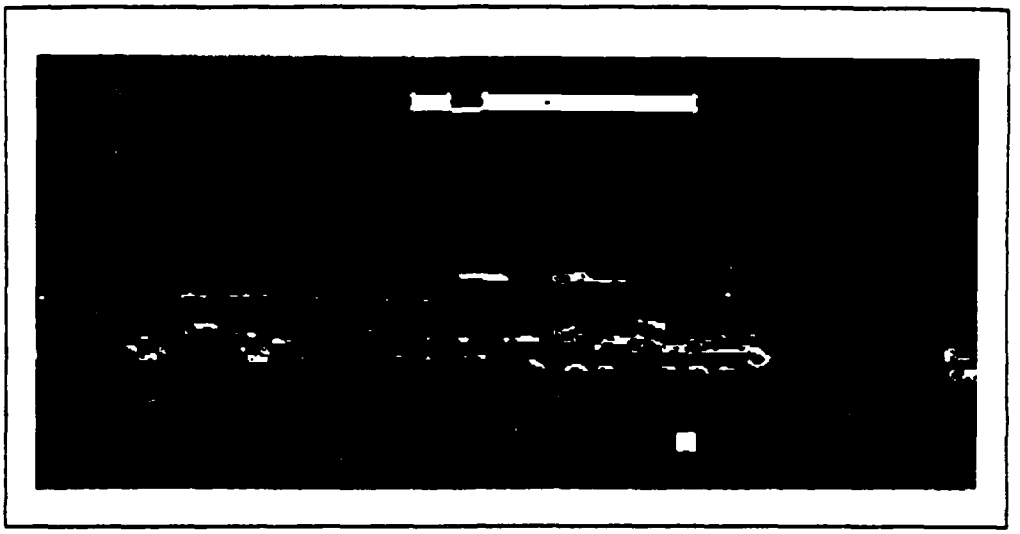


Figure 4.5b. Cosine Gabor transform of modified FLIR image REFM13.FLR. Note that targets are still indicated even with the high intensity corruption introduced by the blocks.

This intensity dependence is not as detrimental if the image possesses little to no correlation with the frequency and the orientation of the distribution represented by the cosine wave Gabor function. There is significant correlation with the 45 degree orientation Gabor and the 135 degree orientation Gabor functions which were added to REFM13.FLR. This result is expected since the 3/16 cycles per pixel and the orientations exactly match.

The effects of using sine wave Gabor functions for image segmentation is shown in Figure 4.6.

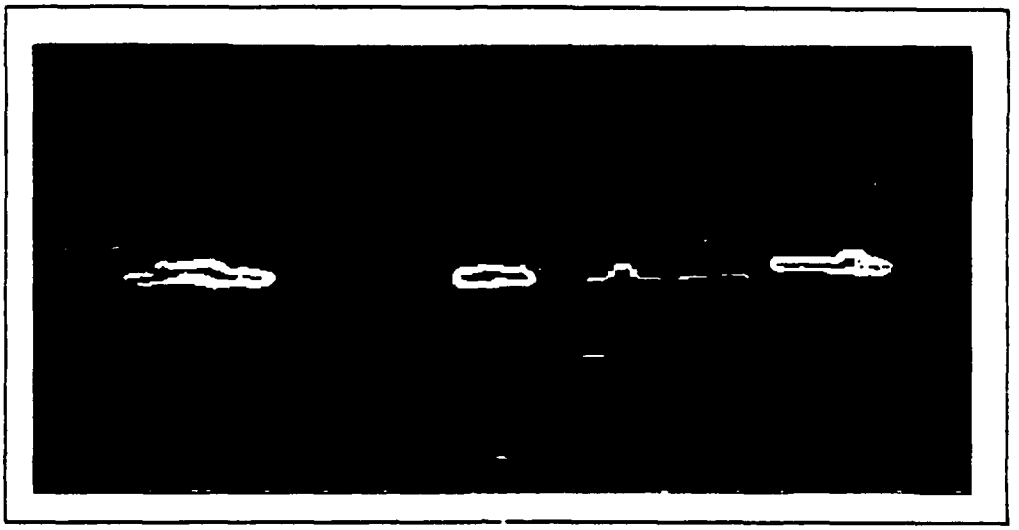


Figure 4.6 Binarized version of sine wave Gabor transform of FLIR image REFJ15.FLR. Sine wave Gabor functions act as edge detectors.

The sine wave Gabor function primarily detects edges provided the particular sine wave frequency distribution has sufficient correlation with the frequency distribution present within the raw FLIR image.

The sine wave Gabor function acts as an edge finder, disregarding those regions with little changes relative to the frequency and orientation of the sine wave Gabor function, finding only those regions within a given image that correspond to a particular frequency and in-plane rotation (orientation).

The sine wave Gabor function (and the cosine wave Gabor function also for that matter) will reject regions within a given

image possessing noise provided the noise does not possess sufficient correlation with the spatial frequency and orientation of the Gabor functions.

Therefore, if differences exist within a given image, and those differences correlate with a particular frequency and spatial orientation of the sine wave Gabor function, then an indication will result in that region within the image.

It should be noted that the sine Gabor and the cosine Gabor are subject to the overall average intensity found in the FLIR image. If the intensity of a target is relatively close to the average intensity within the image, then the Gabor functions may be hard pressed to segment those targets. In other words, if the intensity differences within a region are not sufficiently different, then neither the sine nor the cosine Gabor function will discriminate. The targets in the 113 FLIR images used for this research possessed enough intensity to prevent this problem. It may be necessary to Lambertize an image prior to Gabor transformation if the targets have intensities at or near the overall image average intensity.

Specific discussion. A specific discussion is provided to give the reader in-depth information on the effects of Gabor transformation on FLIR imagery.

Frequency variation.

Non-Self Similar versus Self Similar. For purposes of illustration, sine wave frequency variation will be discussed. Figures 4.7a, b, and c show the effects of increasing the sine variation on the superposition of correlations between raw FLIR image RRFJ15.FLR and non-self similar Gabor functions of orientation 0, 45, 90, and 135 degrees, size 16x16. Figure 4.7a is 1/16 cycles per pixel, figure 4.7b is 3/16 cycles per pixel, and figure 4.7c is 5/16 cycles per pixel.

As the sine wave frequency increases, higher and higher frequency content within the image is exposed. Edges become more and more enhanced. However, as the frequency increases beyond 5/16 cycles per pixel, the higher frequency noise in resulting Gabor filtered image begins to corrupt the segmentation.

Apparently, the raw FLIR image data used for this research have an optimum frequency range between 3/16 cycles per pixel and 5/16 cycles per pixel. Optimum is defined as the best segmenting of objects from background as discerned by the human visual system.

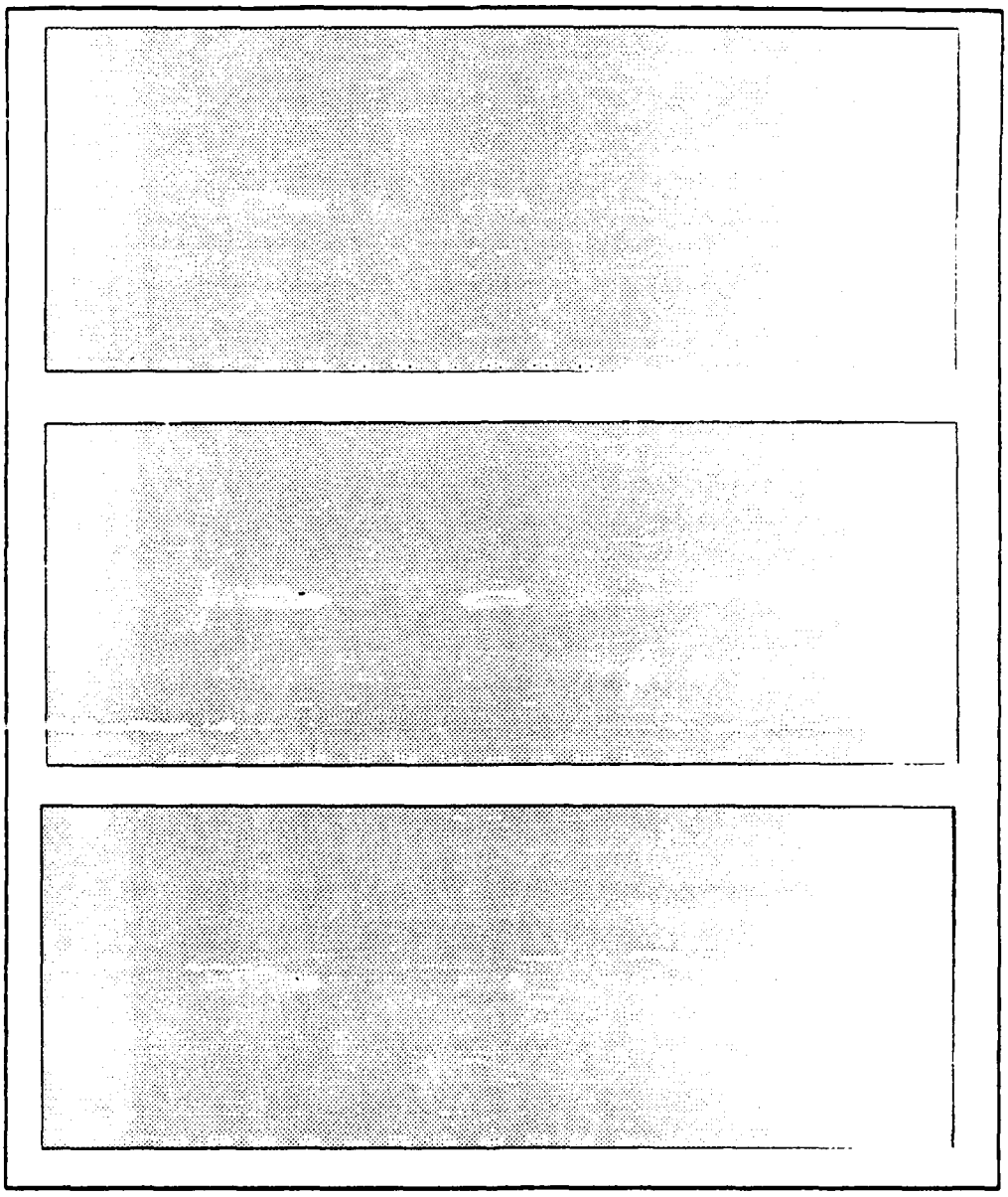


Figure 4.7a/b/c. Effects of changing the frequency of the non-self similar sine Gabor sinusoidal cofactor. As the frequency increases, the correlations become more intense until an "optimum" frequency is reached.

Furthermore, the frequency distributions within the image at 3/16 cycles per pixel tend to be oriented at 45 and 135 degrees from vertical, whereas, at 4/16 cycles per pixel and 5/16 cycles per pixel, the orientation of the frequency distribution tends to be vertical and horizontal. This phenomenon shows the keen sensitivity of the Gabor functions to slight changes in frequency distributions or textures. Hence, the Gabor functions are very useful as texture discriminators and can segment FLIR images based on this characteristic.

Figures 4.8a, b, and c show the effects of increasing frequency on the superposition of correlations between raw FLIR image RPFJ15.FLR and self similar Gabor functions of orientation 0, 45, 90, and 135 degrees, Gabor size 16x16. By maintaining a 16 by 16 Gabor patch size, the dilations of the self similar Gabor function simply change the correspondence between the maximum extent of the Gabor function and the standard deviation at that extent. One cycle within the 16 pixel extent corresponds to one standard deviation at the maximum extent of the 16x16 Gabor patch. Two cycles per 16 pixels corresponds to two standard deviations at the maximum extent and so forth up to 5 cycles per 16 pixels.

Figure 4.8a corresponds to one standard deviation at the maximum extent of the Gabor function (16x16), figure 4.8b

corresponds to 3 standard deviations, and figure 4.8c corresponds to 5 standard deviations.

As the frequency of the sine wave variation increases, the higher and higher frequency content becomes more and more apparent. However, the self similar Gabor functions tend to "double image" the edges of the objects at 3/16 cycles per pixel (figure 4.8a).

Since the standard deviation of the Gaussian envelope is allowed to vary, more energy is included in the correlation between the image and the particular Gabor functions than is found in the non-self similar Gabor functions. Therefore, one would expect a more significant ringing effect at this frequency. As the frequency increases, and consequently, the spread of the Gaussian cofactor decreases, more localization of energy will occur. This effect is desirable for segmentation since the premise for segmentation is the localization of energy into regions of interest. However, when using the conventional self similar Gabor transform, the frequency of the sinusoidal cofactor must match the frequency resident in the image. Since the frequency of the sinusoidal cofactor is equivalent to the Gaussian standard deviation, localization of energy must also be sufficient to segment. Figures 4.8b and 4.8c show significant localization but little to no frequency "matching". Hence, less and less of the target regions correlate as the frequency increases (standard deviation decreases).

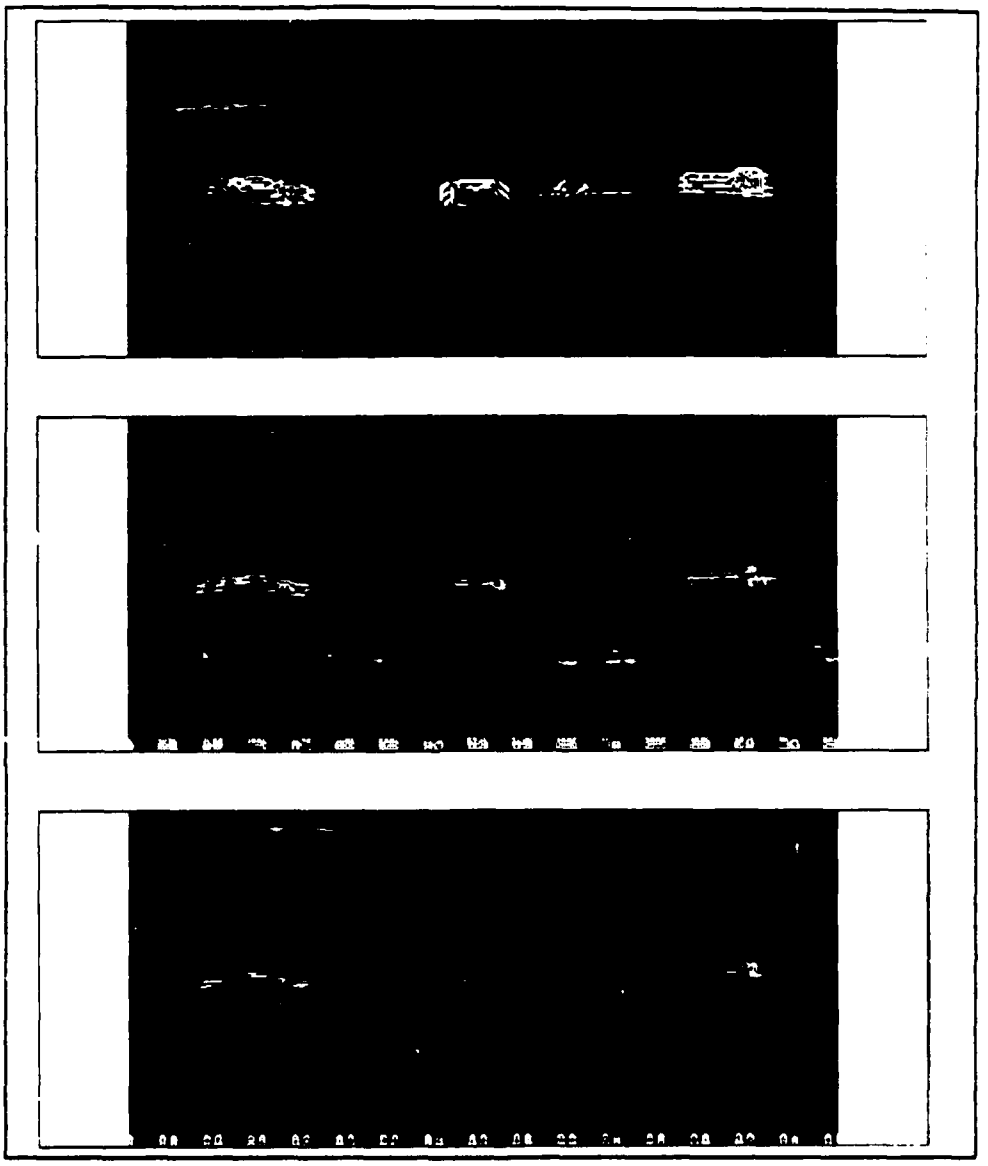


Figure 4.8a/b/c. Effects of changing the frequency of the self similar sine Gabor sinusoidal cofactor. As the frequency increases, the spread decreases, hence more localization of correlation energy.

Cosine versus Sine. For the purpose of discussion, non-self similar Gabor functions will be used for the section. Figures 4.9a, b, and c show the effects of increasing the cosine variation on the superposition of correlations between raw FLIR image REFJ15.FLR and Gabor functions at orientation 0, 45, 90, and 135. Figure 4.9a corresponds to 1/16 cycles per pixel, figure 4.9b corresponds to 3/16 cycles per pixel, and figure 4.9c corresponds to 5/16 cycles per pixel.

As the frequency increases, the correlation energy is spread into the higher order lobes of the Gabor function thus giving the cosine Gabor function some insensitivity to gross intensity, a feature inherent to sine wave Gabor functions.

In figure 4.9a, the near replication of the original image is apparent. The region corresponding to the tank (far left center) is beginning to take shape, although the front of the tank is somewhat obscured. The armored personnel carrier and truck are readily discernable. However, the target board is also obscured.

Figure 4.9b shows shapes forming which correspond to the tank (far left center), armored personnel carrier (center), target board (right center), and truck (far right center). Again, the image represented in Figure 4.9c is a near replication of the original image (Figure 4.4a).

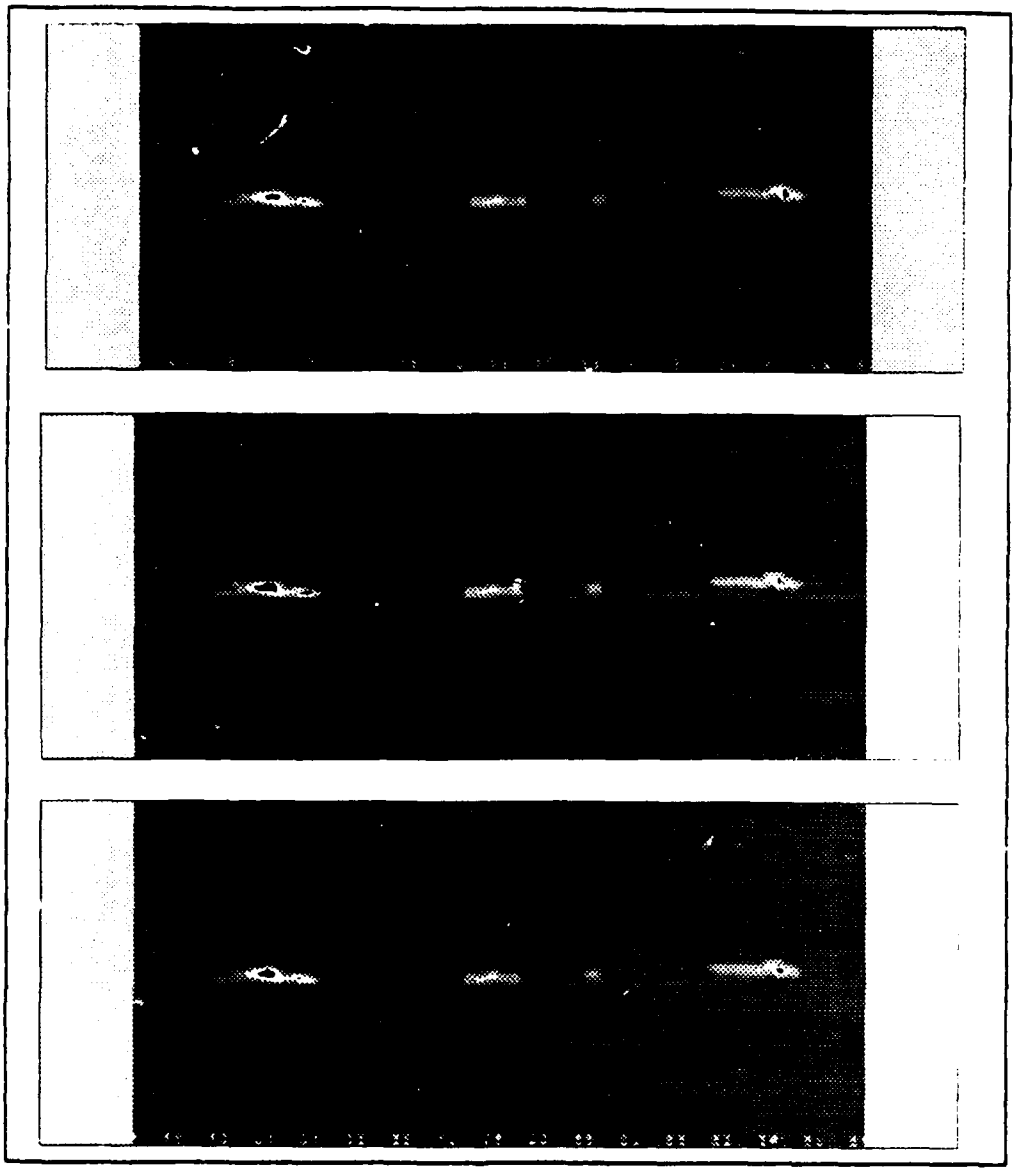


Figure 4.9a/b/c. Effects of changing the frequency of the non-self similar cosine Gabor sinusoidal cofactor. As the frequency increases, target indications become more readily apparent until an "optimum" frequency is reached.

Figure 4.9c shows the effect of increasing the cosine frequency to 5/16 cycles per pixel. The regions corresponding to tank, armored personnel carrier, and truck are readily seen. However, the target board between the armored personnel carrier and the truck is obscured. This may be due, in part, to the cosine dependency on relative intensity within the target board region as well as inappropriate frequency correlation. The background is nearly eliminated, leaving only the targets within the image.

The lighter regions which borders the left side of this series of images is due to the 512 pixel limitation imposed by the FFT subroutine but does not affect the performance of the Gabor transform to segment image. The "tick" marks along the base of the images are artifacts generated by the original image processing.

This series of images show the increased necessity for having proper frequency and orientation matching. The higher intensity and body shaping found in Figure 4.9c is a striking example of this requirement.

Gabor Size.

The size of the Gabor function, namely the spread, or standard deviation, works to isolate or localize the correlated energy into regions specific to edges (when using the sine wave Gabor function) and bodies (when using the cosine Gabor function).

As the Gabor size becomes larger, more and more energy beyond the edges is being correlated with the Gabor function, acting to diffuse the edges.

This concept is similar to that proposed by Siebert and Waxman in their paper about a neural network design which, essentially, diffuses energy from regions surrounding edges towards the edge to localize the detection of edges: segmentation (22:9-27). The Gabor function works to localize the energy by reducing the extent of the spreading via the Gaussian envelope and retains frequency specificity by introducing the sinusoidal variation. Hence, the Gabor function produces a more localized correlation while maintaining a specific "matched" filter characteristic.

Non-Self Similar versus Self Similar. The sequence of figures 4.10a, b, and c shows the effect of increasing the Gabor size yet maintaining the cycles per pixel ratio (non-self similar Gabor "patch" size).

Figure 4.10a is the superposition of the correlations between non-self similar Gabor functions of "patch" size 64x64, frequency 32, orientations 0, 45, 90, and 135 degrees with the raw FLIR image REPM13.FLR (four images superimposed).

Figure 4.10b is the superposition of the correlations between non-self similar Gabor functions of "patch" size 32x32, frequency

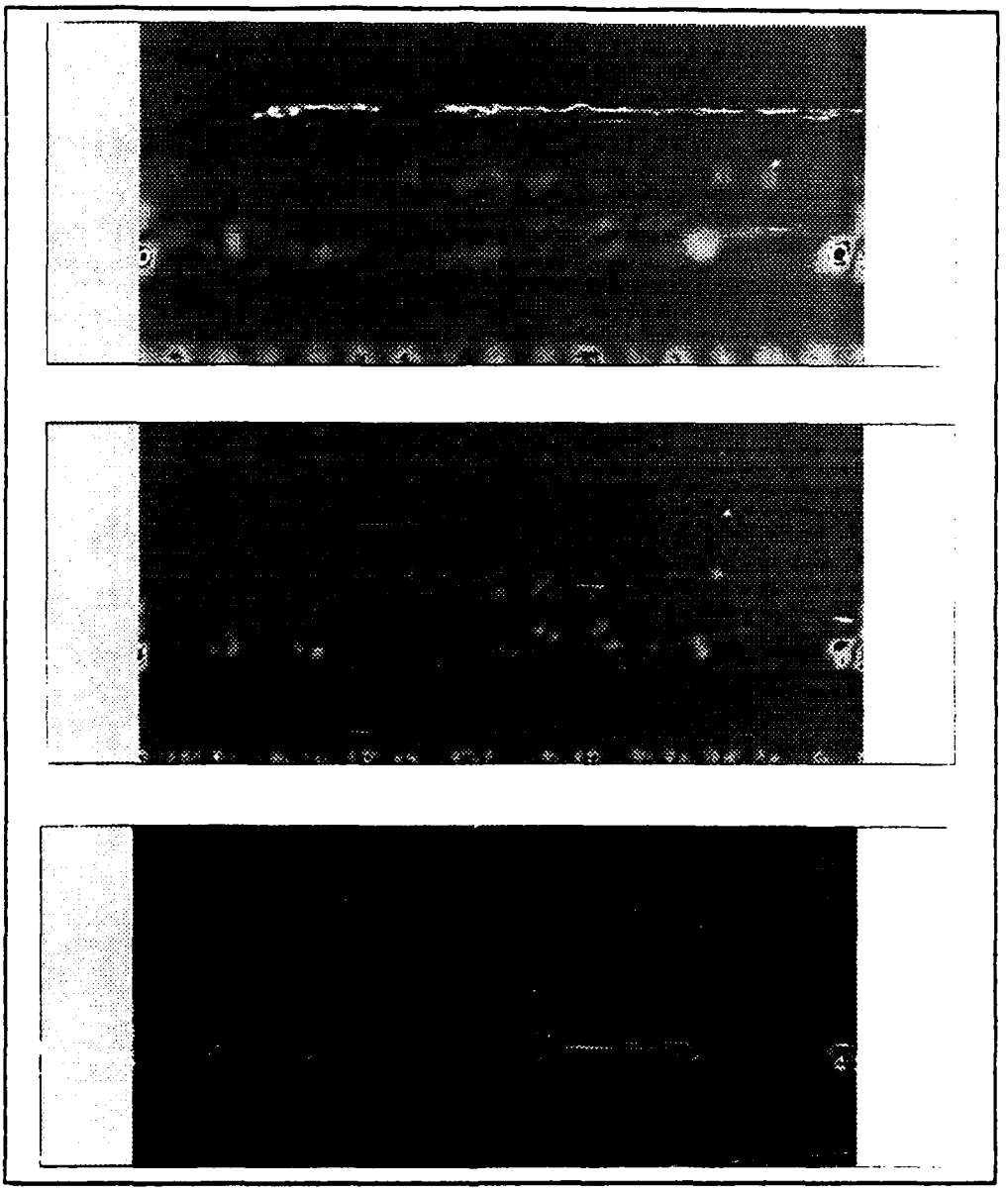


Figure 4.10a/b/c. Effects of changing the Gabor patch size on non-self similar Gabor transformation of FLIR image REFM13.FLR. As the Gabor patch size increases, blurring occurs due to increase in correlation area.

16, orientations 0, 45, 90, and 135 degrees with the raw FLIR image REPM13.PLR.

Figure 4.10c is the superposition of the correlations between non-self similar Gabor functions of "patch" size 8x8, frequency 4, orientations 0, 45, 90, and 135 degrees with the raw FLIR image REPM13.PLR.

When the Gabor size changes, with the number of cycles remaining the same, the effect is that of a self similar Gabor function. The standard deviation of the Gabor functions changes as the Gabor size dilates. The frequency changes proportionally to the Gabor size as well. This effect is, by definition, self similar. The frequency, therefore, must increase as the Gabor size increases to maintain a constant cycles per pixel ratio and to retain the non-self similar nature of the Gabor function.

As the Gabor patch size increases, more energy from the background surrounding the target within the image is correlated with the Gabor function, hence, less localization of energy about the target(s) within the image. Consequently, as the Gabor patch size increases, the "resolution" of the filtered image is decreased and the ability of the Gabor function to segment the image is diminished.

In figure 4.10a, the detected regions tend to be "blotchy", no coherent form to the regions of relatively high correlation.

This appears to be a "de-focused" view of the regions surrounding the front viewed tank, foreground, side viewed tank, foreground, and truck, foreground. No discernable targets can be seen in the background.

Figure 4.10b shows more localized correlations, localized to the regions of interest, namely targets in foreground and background. However, no recognizable shapes can be seen. Again, this image appears to be a "de-focused" view of the targets.

Figure 4.10c shows very localized correlation energy. Outlines of the treads of the front viewed tank, foreground are readily apparent. The shape of the side viewed tank is beginning to take form. As is seen in figure 4.10a and 10b, the lighter region to the left of the image is due to the 512 pixel limitation of the FFT subroutine.

The sequence of figures 4.11a, b, and c shows the effect of increasing the window size while maintaining one cycle per envelope (self similar Gabor "patch" size).

Figure 4.11a is the superposition of the correlations between self similar Gabor functions of "patch" size 64x64, frequency 1, orientations 0, 45, 90, and 135 degrees with the raw FLIR image REFM13.FLR (four images superimposed).

Figure 4.11b is the superposition of the correlations between self similar Gabor functions of "patch" size 32x32, frequency 1,

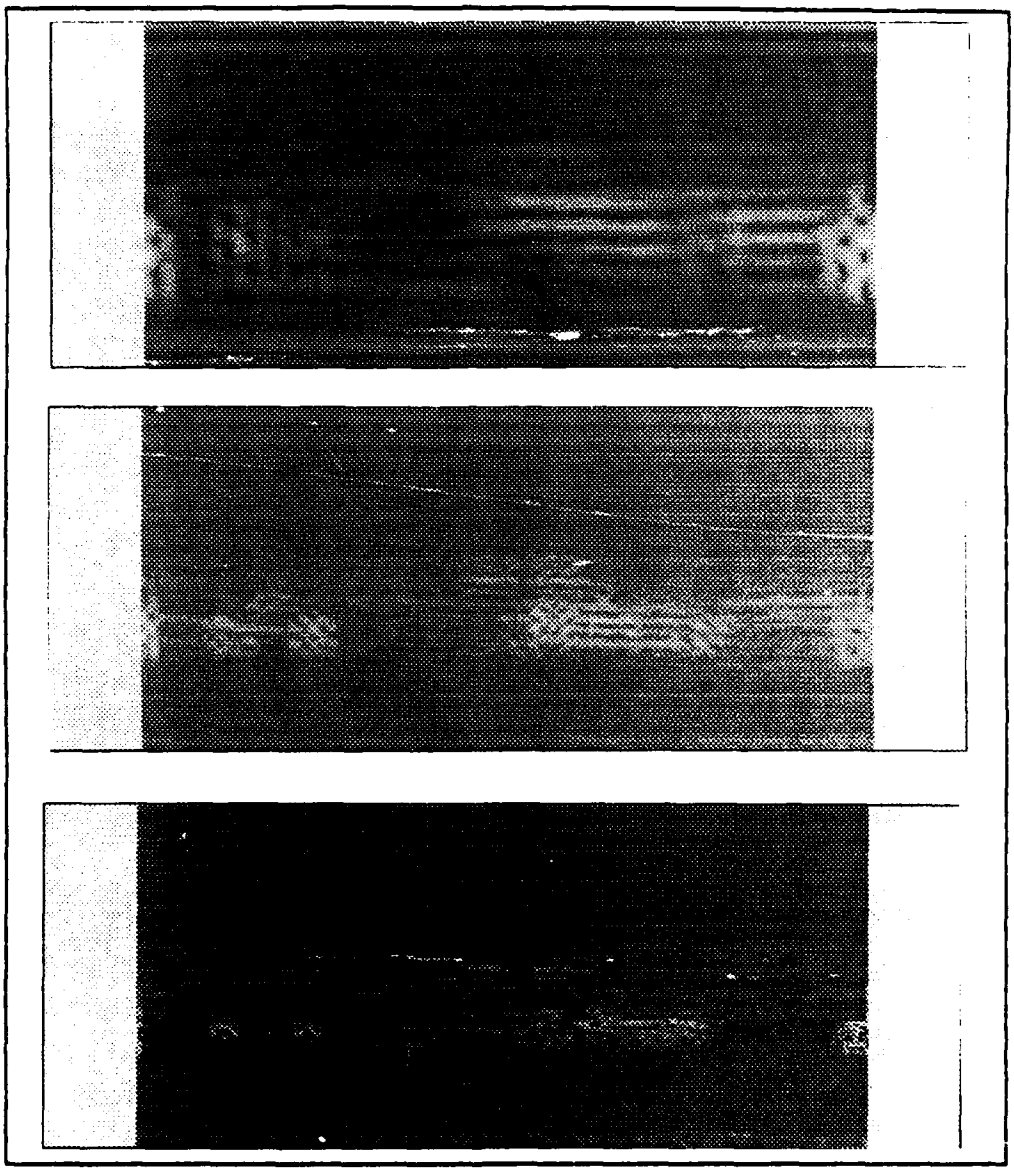


Figure 4.11a/b/c. Effect of changing Gabor patch size on self similar Gabor transformation of FLIR image REFM13.FLR. As the Gabor patch size increases, more image pixels are included in the correlation region, hence blurring occurs.

orientations 0, 45, 90, and 135 degrees with the raw FLIR image REFM13.FLR.

Figure 4.11c is the superposition of the correlations between self similar Gabor functions of "patch" size 8x8, frequency 1, orientations 0, 45, 90, and 135 degrees with the raw FLIR image REFM13.FLR.

By keeping the frequency constant but varying the window size, the net effect is that of self similarity. As the window size increases, the Gabor function frequency decreases. When the window size increases, the frequency of the Gabor function decreases, which "blurs" the image. Conversely, making the window size smaller, hence increasing the frequency of variation of the Gabor function, will tend to "sharpen" the image relative to a larger window size.

The Gabor window size is important for energy localization. If the Gabor window is too large, then the ratio between the background energy and target energy within the correlation region is high and the correlation is primarily with the background and not the target. In other words, the energy is no longer localized on and immediately around the target, rather spread or diffused into the background region surrounding the target, blurring the target region. This effect is clearly evident in the series of images (4.11a, b, and c sequence).

Figure 4.11c shows the effect of a large window size (64 by 64). The regions surrounding the targets are indistinguishable. No shapes can be seen. Although, in a global sense, the energy is beginning to be localized to the target regions.

Figure 4.11b shows more localization due to the decrease of window size to 32 by 32. The region corresponding to front viewed tank, foreground and side viewed tank, foreground are significantly localized suggesting that self similar Gabor functions can, in fact, segment (see Appendix M).

Figure 4.11c shows the effect of decreasing the window size to 8 by 8. Energy is localized to regions surrounding the targets in the background. However, the corresponding frequency is high enough to detect the clutter surrounding the targets in the foreground.

The results in figures 4.11a, b, and c suggest that care must be taken to ensure the interplay between localization and frequency match is optimized to use self similar Gabor functions for segmentation.

In summary, Gabor transforms successfully segment the FLIR images used in this research. Sine wave Gabor functions act as edge detectors whereas cosine Gabor functions act as "body fillers". In combination, sine and cosine Gabor functions produce fully segmented targets. Since the segmentation of FLIR images is successful, then,

from a biological standpoint, the metric for segmentation should also be the metric for classifying or recognizing.

Part 2. CLASSIFICATION

Part 2 of chapter IV discusses the preliminary investigation into the use of Flaton and Toborg's jet vector approach for classification (5:I313-I320). Several approaches were investigated. The first attempt to classify targets based on Gabor coefficients used five layers (corresponding to frequencies), each with four orientations (0, 45, 90, and 135) to comprise a twenty dimensional jet vector. The second attempt reduced the number of layer to two, retaining the four orientation (sublayers) to produce eight dimensional jet vectors. The third attempt reduced the number of possible jet vectors to five specifically located jets. The fourth attempt employed a kth nearest neighbor network to investigate variations in number of possible classes and number of input components.

Twenty Dimensional Jet Vector Approach.

Requirements for Classification. Based on the results of using non-self similar Gabor functions to segment FLIR images, certain requirements were needed to investigate a Gabor based classification system.

These requirements were 1) jet vectors with components having a "meaningful" value, 2) properly thresholded images to produce consistent jet locations, and 3) sufficient dynamic range for vector components to discriminate between vectors.

"Meaningful vector component value". The concept of "meaningful" vector component value is simply stated: If the value for the correlation coefficient is derived from primarily target pixels and not the background, then the component of the jet vector is "meaningful" relative to gaining information about the target.

When the background occupies more than 50 percent of the Gabor window size, then the Gabor coefficient value associated with that jet vector component does not convey information about the target, rather it conveys information about the background.

For the FLIR images used in this research, a Gabor window size 32 by 32 or greater did, in general, exceed the 50 percent limit. Therefore, the twenty dimensional jet vector approach for classification did not produce "meaningful" vector component values since twelve out of the twenty components for each vector had values corresponding to the correlation with the background and not with the target.

Classification of targets into truck, tank, and APC for FLIR images in Data Set2 (Exemplar set from REFM13.FLR, Comparison sets from REFM14.FLR through REFM19.FLR) based on twenty dimensional vectors was not successful. Less than 5% of the targets were properly identified. This was not unexpected since most of the component values of the vectors were correlation coefficients associated with the background and not with the target.

Proper thresholding for consistent jet locations. Another problem with the twenty vector approach was the on center nature of the vectors used for exemplars. For instance, Gabor filtering FLIR image, REFM13.FLR using frequency 3/16 cycles per pixel, orientations 45 and 135 degrees for sine wave Gabors, orientations 0 and 90 degrees for cosine wave Gabors, only six on center locations were produced.

The locations were associated with the left and right treads of the front-viewed tank in the foreground, the body of the side viewed tank in the foreground, the rear cog of the side viewed tank in the foreground, the barrel flashguard on the side viewed tank in the foreground, and the truck in the background.

Apparently, the thresholding of the image eliminated the armored personnel carrier in the background as well as the finer detailed features of the side viewed tank in the foreground, such as the road wheels and upper rollers, features which could be used for classification.

With proper thresholding, the finer details of the targets within the image(s) will be passed, thus providing multiple looks within a given target region.

In hindsight, classifying targets based on a vector located at the center of a target did not make sense. This approach is analogous to a psychophysical test of "flashing" an image in front

of a subject for less than 100 milliseconds and then asking the subject to describe the entire image. Since humans cannot begin scanning an image in less than 100 msec, only a single look is possible (13:67).

The assumption that enough information may be at the center of each object to classify that object is risky. Consider describing an elephant by looking at a 1/8 inch patch, on center along the side of the elephant.

Therefore, multiple looks within a target region combined with limiting the Gabor window to 16 by 16 and 8 by 8 was considered a viable alternative to the single, on center look with window sizes ranging from 128 by 128 to 8 by 8, reduction in size by octaves.

Sufficient dynamic range of vector components. Typical values for vector components using the eight component vector approach ranged from 0.5 to 1 which significantly reduced the dynamic range for vector distances. By "spreading" the vector component values to range from zero to unity using equation 8, the dynamic range for distances was increased to produce sufficient distances between vectors, thus enhancing the chance for classification based on this feature.

Eight Dimensional Jet Vector Approach.

Data Set1 (Exemplar vectors from REFM13.FLR). After proper segmentation and threshoiding of FLIR image REFM13.FLR, twenty-three, eight dimensional vector exemplars were generated instead of six twenty dimensional vector exemplars. The twenty-three vector exemplar set was compared to the test vectors from FLIR images REFM14.FLR through REF..19.FLR.

The attempt to classify targets using eight component vectors (Gabor coefficients from 16 by 16 and 8 by 8 window sizes) was not successful. Table 2a shows typical results from determining the Euclidean distance between exemplar (REFM13.FLR) and test vectors (REFM17.FLR, specifically).

REFM17.FLR possessed 24 locations from which jet vectors were constructed. Each column corresponds to the vector location. For instance, columns 1 through 8 correspond to locations 1 through 8 in FLIR image REFM17.FLR. The first six numbers in each column represent the six shortest distances between the exemplar vectors generated from REFM13.FLR and the specific test vector generated from REFM17.FLR, rank ordered. The last six values in each column correspond to the number of the exemplar vector to which the test vector compared, also rank ordered. For instance, 0.8495 is the distance between exemplar vector 3 and test vector 1, 0.8539 is the distance between exemplar vector 22 and test vector 1, and so forth.

Table 2a. Euclidean distances between exemplar and test vectors (Exemplar: REFM13.FLR, Test: REFM17.FLR)

Test. No.	1	2	3	4	5	6	7	8	9	10	11	12
	0.850	0.294	0.653	0.431	0.764	0.902	0.616	0.599	0.654	0.150	0.506	0.506
	0.854	0.527	0.733	0.526	0.764	0.964	0.632	0.638	0.891	0.635	0.730	0.562
Eucl.	1.129	0.585	0.769	0.754	0.786	1.143	0.752	0.754	0.986	0.708	0.873	0.592
Dist.	1.136	0.655	0.779	0.835	0.868	1.227	0.796	0.756	1.038	0.865	0.905	0.639
	1.156	0.712	0.822	0.838	1.015	1.237	0.855	0.771	1.038	0.925	0.914	0.641
	1.227	0.815	0.989	0.925	1.038	1.334	0.913	1.026	1.046	0.945	0.971	0.743
	3	6	7	8	8	3	6	6	13	7	6	13
Tmpl	22	2	2	10	22	4	7	2	14	16	2	14
No.	11	1	6	16	4	10	1	1	22	15	1	6
	4	19	1	9	10	22	16	7	1	6	8	19
	15	12	16	14	16	9	2	12	19	2	7	2
	18	13	15	4	9	8	8	19	4	12	12	23

Test. No.	13	14	15	16	17	18	19	20	21	22	23	24
	0.417	0.382	0.477	0.922	0.679	0.337	0.573	0.718	0.434	1.227	1.579	0.866
	0.679	0.508	0.477	1.180	0.794	0.382	0.767	0.867	0.655	1.311	1.596	1.140
Eucl.	0.925	0.618	0.653	1.271	0.886	0.602	0.798	1.007	0.754	1.398	1.772	1.191
Dist.	0.945	1.038	0.919	1.291	1.123	0.797	0.980	1.017	0.911	1.399	1.790	1.241
	0.950	1.083	1.059	1.311	1.230	0.826	1.012	1.102	0.945	1.452	1.794	1.338
	0.968	1.217	1.097	1.311	1.254	0.978	1.078	1.121	1.116	1.462	1.810	1.349
	15	15	15	3	16	10	11	22	19	19	21	3
Tmpl	11	7	7	4	15	8	3	3	1	1	1	21
No.	18	16	16	22	7	14	18	11	23	3	3	4
	12	12	12	19	8	13	15	4	2	23	20	22
	16	11	6	10	1	4	17	9	6	17	6	15
	22	6	1	1	11	6	5	10	13	10	15	1

Table 2b. Figure of Merit (F.O.M.) associated with data in Table 2a.

	1	2	3	4	5	6	7	8	9	10	11	12
2	1.005	1.789	1.122	1.220	1.000	1.069	1.025	1.065	1.364	4.218	1.442	1.109
3	1.329	1.987	1.177	1.749	1.029	1.268	1.220	1.259	1.508	4.706	1.724	1.170
4	1.337	2.226	1.192	1.935	1.135	1.361	1.292	1.263	1.587	5.747	1.788	1.262
5	1.361	2.421	1.258	1.942	1.328	1.373	1.387	1.287	1.588	6.147	1.805	1.266
6	1.445	2.768	1.513	2.144	1.358	1.479	1.481	1.714	1.600	6.289	1.918	1.466

	13	14	15	16	17	18	19	20	21	22	23	24
2	1.628	1.329	1.000	1.280	1.170	1.132	1.340	1.208	1.511	1.068	1.015	1.315
3	2.216	1.615	1.370	1.378	1.306	1.784	1.393	1.402	1.739	1.140	1.127	1.375
4	2.265	2.715	1.928	1.400	1.655	2.363	1.713	1.419	2.100	1.140	1.138	1.432
5	2.276	2.832	2.202	1.422	1.812	2.449	1.767	1.534	2.179	1.184	1.141	1.544
6	2.320	3.183	2.300	1.422	1.848	2.900	1.882	1.560	2.573	1.191	1.151	1.557

Table 2b shows the figure of merit assigned to the distances within a given column relative to the minimum distance. Test vectors 20 through 23 were generated at locations corresponding to road wheels associated with the side viewed tank in the foreground. Each of these vectors were correctly matched with vectors from the test set. However, the figure of merit associated with these "correct" responses ranged from 1.0681 to 1.5105.

Test vector ten, which was generated at a location associated with the right tread of the front viewed tank in the foreground in

FLIR image REFM17.FLR, did not match with the choices given. The figure of merit suggests, however, that the first choice, exemplar vector 7 (associated with the front end of the side viewed tank in the foreground), was a very good choice, F.O.M. equal to 4.2182.

These results indicate an inconsistency in generating a feature for classifying targets. It should be noted that these results are based on a very limited data set. This inconsistency was not unexpected considering the segmentation centroid location scheme.

First, segmentation of the REFM series of FLIR images was accomplished using sine wave Gabor function. Since sine wave Gabor functions produce edge enhanced targets within the image, subsequent thresholding and blob centroid determination produced jet vector locations at the periphery of the targets within the image.

Second, by generating vector locations along the periphery of the targets within the FLIR image, roughly half of the Gabor window correlated with the target and half with the background. Therefore, successful target classification is directly attributable to two factors: 1) the clutter (noise) surrounding the target(s) and 2) the amount of "meaningful information" within the target(s) at the vector location(s). Meaningful information, in this application, is meant to imply that, at the specific vector location(s), there is enough information about the target to classify it.

Only 11.7 percent of the target components (treads, wheel cogs, hull, turret, etc.) were correctly classified using this approach. Another method was attempted for classifying detailed structure(s) within the target regions. This alternative method involved limiting the construction of the eight dimensional jets to five, manually and specifically placed locations.

Data Set2 (Exemplar vectors from REFM14.PLR). Table 3a shows typical results for determining the distance between the exemplar (REFM14.PLR) and test set (REFM13.PLR) jet vectors specifically located within the foreground target regions. Each column corresponds to the specified location in the test vector set: 1) Mid Hull, side viewed tank, foreground, 2) Rear cog, side viewed tank, foreground, 3) Front cog, side viewed tank, foreground, 4) Left tread, front viewed tank, foreground, 5) Right tread, front viewed tank, foreground.

The first five values in each column correspond to the actual distances, rank ordered from one to five. The last five values in each column correspond to the exemplar vector number, as specified above. For instance, the distance 0.7478, in column one, corresponds to the distance between test vector one and exemplar vector one, and so on. Therefore, for a perfect classification score, row six of the ten by five matrix should read 1, 2, 3, 4, 5.

Table 3a. Euclidean distances between exemplar and test vectors (Exemplar: REFM14.FLR, Test: REFM13.FLR)

	COLUMN 1 MID-HULL	COLUMN 2 REAR COG	COLUMN 3 FRNT COG	COLUMN 4 LEFT TR.	COLUMN 5 RGHT TR.
DIST	0.7478	0.9677	0.8373	0.9011	0.7593
	1.2410	1.4621	1.2049	0.9013	1.0107
	1.2764	1.4974	1.2115	1.0340	1.0740
	1.5588	1.5025	1.5434	1.4700	1.2724
	1.5913	1.5177	1.6102	1.6519	1.4516
TGT	1 MID-HULL	2 REAR COG	2 REAR COG	4 LEFT TR.	3 FRNT COG
	2 REAR COG	4 LEFT TR.	3 FRNT COG	5 RGHT TR.	5 RGHT TR.
	5 RGHT TR.	5 RGHT TR.	1 MID-HULL	3 FRNT COG	4 LEFT TR.
	3 FRNT COG	1 MID-HULL	5 RGHT TR.	2 REAR COG	2 REAR COG
	4 LEFT TR.	3 FRNT COG	4 LEFT TR.	1 MID-HULL	1 MID-HULL

The mid hull, side viewed tank, foreground vector of the test set matched with the mid hull, side viewed tank, foreground vector of the exemplar set, as did the rear cog, side viewed tank, foreground and the left treat, front viewed tank, foreground vectors. The front cog, side viewed tank, foreground vector and the right tread, front viewed tank, foreground vector were second choices when comparing REFM13.FLR to REFM14.FLR. This raises the question, " How good is first, second, third choices ? ".

Table 3b shows the figure of merit associated with second through fifth choices for comparison between exemplar (REFM14.FLR) and test set (REFM13.FLR).

Figure of merit values greater than 1.35 usually indicate strong confidence of choice. Therefore, the first choices for vectors one, two, and three have strong confidence. However, the figure of merit values for vectors four and five are below 1.35. In particular, vector 4 has a F.O.M. of 1.0002, indicating very low confidence in its first choice, suggesting that its second choice is equally probable.

These typical results indicate promise in using Gabor correlation coefficients for classifying targets within FLIR images used in this research. This method is not fault-proof as is readily apparent in the data presented in tables 3a and 3b. The second choice for vector three is the correct choice yet its F.O.M. is

Table 3b. Figur- of Merit (F.O.M.) associated with data in Table 3a.

	COLUMN 1 MID-HULL	COLUMN 2 REAR COG	COLUMN 3 FRNT COG	COLUMN 4 LEFT TR.	COLUMN 5 RGHT TR.
2	1.6594	1.5109	1.4390	1.0002	1.3312
3	1.7068	1.5474	1.4469	1.1474	1.4145
4	2.0844	1.5526	1.8432	1.6314	1.6759
5	2.1279	1.5684	1.9231	1.8332	1.9118

1.439, well above the 1.35 threshold for high confidence. The second choice for vector five has a F.O.M of 1.3312, slightly under the 1.35 high confidence threshold, yet this choice is incorrect.

Table 3c shows the overall effects of limiting the construction of jets to the five specific locations. The first column (heading TEST) indicates the image from which the test vectors were derived. "Target number" indicates the view of the targets in the foreground (1 corresponds to side viewed tank, 2 corresponds to front viewed tank).

Table 3c. Performance of jet vector recognition using five specifically located jets.

Test	Tgt #	On Target	Detailed Substructure	Test	Tgt #	On Target	Detailed Substructure
M13	1	100/67/33	60/40/0	K30	1	100/33/100	40/40/20
	2	50/100/50			2	100/50/50	
M15	1	67/33/33	40/20/0	K31	1	67/0/33	40/0/20
	2	100/100/0			2	50/50/0	
M16	1	33/33/33	20/40/20	K32	1	100/0/67	60/20/0
	2	100/100/0			2	100/50/0	
M17	1	100/67/67	60/0/20	K33	1	33/0/100	20/20/20
	2	50/100/50			2	50/50/0	
M18	1	67/67/67	20/40/20	K34	1	33/0/67	0/40/0
	2	50/100/50			2	50/100/0	
M19	1	67/33/67	20/20/20	K35	1	33/67/100	0/40/20
	2	100/50/50			2	50/50/0	
K19	1	67/33/67	20/20/0	K36	1	33/33/100	0/40/20
	2	50/100/0			2	0/100/50	
K20	1	33/33/100	20/40/0	K41	1	67/0/67	20/40/0
	2	50/100/50			2	100/100/0	
K21	1	33/67/33	0/60/0	K42	1	33/67/67	20/40/40
	2	50/100/0			2	50/50/50	
K22	1	33/100/33	20/60/0	K43	1	33/0/100	0/20/0
	2	100/50/50			2	100/50/0	
K23	1	0/0/100	0/0/20	K44	1	100/0/67	20/20/20
	2	0/50/0			2	50/50/0	
K24	1	67/67/67	20/40/0	K45	1	67/0/67	20/20/20
	2	0/50/0			2	50/50/0	
K29	1	67/0/67	40/20/0	K46	1	67/33/33	40/40/0
	2	100/100/0			2	50/100/0	
				K47	1	33/33/67	0/20/40
					2	50/100/50	

The set of three numbers in the "On Target" column indicate percentage of first, second, and third choice correct answer. For example, for test set REFM13.FLR, target 1, the "on target" percentage for first choices is 100 percent, second choices is 67 percent, and third choices is 33 percent.

Note, however, that the correctness of "on target" only indicates that the choices for first, second, and third correspond to the target region and not the specific location within that target.

The last column set of three numbers correspond to percent correct in the first choice, the second choice, and the third choice for the detailed sub-structures. For example, using test set REFM13.FLR, 60 percent of the first choices are correct and 40 percent of the second choices are correct. Consequently, zero percent of the third choices will be correct.

The overall performance of this method to detect correct target is 0.589. This number was calculated by adding the first choice percentages in the "on target" column, dividing by the total number of FLIR images tested (27), then dividing by the maximum possible score (200). The overall performance of this method to detect detailed sub-structures is 0.229. This number was calculated by adding the first choice percentages in the "Detailed

Substructure" column, dividing by the total number of FLIR images tested (27), then dividing by the maximum possible score (200).

These results suggest that classification of targets within cluttered FLIR images is possible using Gabor transform coefficients. However, the detailed substructure of targets may not be classified using this technique. These results also suggest that the Gabor coefficients should be an adequate feature to train a neural network such as a kth-nearest neighbor network.

Kth-Nearest Neighbor Approach.

The kth-nearest neighbor network approach used the same jet vectors as used in the multi-layer perceptron approach. The first nearest neighbor accuracy was 0.2786. The third nearest neighbor accuracy was 0.3500. The fifth nearest neighbor accuracy was 0.3930. The seventh nearest neighbor accuracy was 0.3071. The ninth nearest neighbor accuracy was 0.2786.

These results indicate that the ability of the nearest neighbor network to correctly identify jet vectors is better than chance, but still poor, at best. Although the accuracies calculated are low, insight may be gained by looking at the confusion matrix (the network's choices versus the actual vector classifications).

Table 4 shows the confusion matrix associated with the fifth nearest neighbor calculated classifications and accuracies. Each

Table 4. Confusion Matrix for Fifth Nearest Neighbor Calculations.

		ACTUAL CLASSIFICATION				
		0	1	2	3	4
NETWORK OUTPUT	0	15	3	6	3	2
	1	2	10	11	5	7
	2	4	8	6	3	4
	3	4	5	2	14	6
	4	3	4	3	3	9

column corresponds to the actual classification of the jet vector. Each row corresponds to the network output classification. The values entered into this matrix indicate how many times the network chose a certain class relative to the actual classification of the jet vector.

If the network was perfectly correct, the off diagonal values would be zero and the diagonal values would be maximum. As the values indicate, all but one diagonal value is the maximum.

Class zero were correctly identified 15 times out of 28. Class one vectors were correctly identified 10 times out of 28, but confused for class two vectors 8 times out of 28.

Class two vectors were correctly identified 6 times out of 28, being confused for class one vectors 11 times and class zero vectors 6 times out of 28.

Class three vectors were correctly identified 14 times out of 28, being confused for class one vectors 5 times out of 28.

Class four vectors were correctly identified 9 times out of 28, being confused for class one vectors 7 times out of 28.

Recall, the class zero designation corresponded to the mid hull of the side viewed tank, foreground. Apparently, the jet vector recognition scheme is not readily confused when identifying this particular target substructure.

Class one designation corresponded to rear cog of the side viewed tank, foreground. This type of target substructure can be confused with class two designations, front cog of side viewed tank, foreground. This result is not unexpected since the front cog of a tank is similar in structure to the rear cog of a tank.

Class two designation corresponded to front cog of the side viewed tank, foreground. This type of target substructure is readily confused with the rear cog of the side viewed tank foreground. The confusion with the rear cog is to be expected as previously explained.

Class three designation corresponded to left tread, front viewed tank, foreground. This type of target substructure can be confused for the mid-hull and the rear cog equally likely. This may be explained by the horizontal construction of the tread and its apparent likeness to the horizontal construction of the mid hull. Note also that the rear cog jet vector components encompass not only the rear cog but also part of the rear hull region. Apparently, sufficient correlation with the rear hull may be found in the rear cog jet vectors to provide the confusion indicated for class three designated targets.

Class four designation corresponded to the right tread of the front viewed tank, foreground. Class four targets may be confused as class one targets. The explanation for class three targets hold true for class four targets.

The significant feature of the values found in Table 4 is that the diagonal values are the highest values in the matrix, with one notable exception (values corresponding to class two targets). This trend indicates that, although the overall accuracy was poor,

(0.3930), individual classifications were essentially correct. The confusion between class one (rear cogs) and class two (front cogs) targets can be explained by the similarity in structure between front and rear cogs.

Another problem with this particular application of jet vector classification involves the jet construction, obtaining vector components from Gabor functions that are "too small".

If the Gabor function is significantly smaller than the object its centered on, the values for correlation will become highly susceptible to small changes in position and aspect rotation (out of plane rotation).

The jet vectors used to train and test the multilayer perceptron were comprised of Gabor coefficients, half of which were derived from Gabor functions whose main lobe was considerably smaller than the target of interest. Therefore, half of the jet vector components are highly susceptible to small changes in position and aspect angle.

The results of reducing the input vectors to four components, those corresponding to the 16 by 16 Gabor size correlations (Table 5, row 2) and 8 by 8 (Table 5, row 3), produced nearest neighbor accuracies less than those for the eight inputs (Table 5, row 1).

The 8 by 8 vector components used as input are highly susceptible to slight variations in position and aspect angle, thus

Table 5. Nearest Neighbor Accuracies for eight vector component input, four vector component input (16 by 16), and four vector component input (8 by 8).

	1 NN	3 NN	5 NN	7 NN	9 NN
REG	0.2786	0.3500	0.3930	0.3071	0.2786
SS	0.2143	0.2857	0.3143	0.3000	0.3000
	0.2500	0.3000	0.2786	0.3071	0.3143
MOD	0.3286	0.2786	0.3571	0.3714	0.3571
SS	0.2786	0.3000	0.2643	0.2643	0.2571
	0.2286	0.2000	0.2000	0.2500	0.2357
3 cl.	0.4286	0.4500	0.4643	0.4930	0.4571
2 cl.	0.5786	0.5929	0.6000	0.6214	0.5929

explaining the lower accuracies for nearest neighbor calculations.

Recall, all eight vector components were scaled between zero and one using equation 8. The 16 by 16 vector components were scaled accordingly, thereby, being effected of the 8 by 8 vector components to a greater extent. This may explain the reduction in classification accuracy when using the 16 by 16 vector components as input.

Jet vectors were generated from the correlation coefficients between the FLIR image and the modified Gabor functions. The modified Gabor functions consisted of three cycles per envelope,

orientations 0, 45, 90, and 135 degrees, dilations 16 by 16 and 8 by 8.

Under these conditions, the K-Nearest Neighbor calculations produced accuracies of 0.3286 for the first nearest neighbor, 0.2786 for three nearest neighbors, 0.3571 for five nearest neighbors, 0.3714 for seven nearest neighbors, and 0.3571 for nine nearest neighbors (Table 5, row 4).

Again, reducing the number of inputs to those corresponding to 16 by 16 (Table 5, row 6) or 8 by 8 (Table 5, row 7) produced significantly decreases in accuracies.

In an effort to increase the accuracy of classification, the number of classes was reduced to three (Table 5, row 7) and two (Table 5, row 8). The three class problem reduced the number of classes to mid-hull, side viewed tank, foreground (class 0), cogs, side viewed tank, foreground (class 1), and treads, front viewed tank, foreground (class 2). As is readily evident, by reducing the number of classes, the accuracy of classification increased, ranging from 0.4286 (first nearest neighbor) to 0.4929 (seven nearest neighbors).

The two class problem reduced the classes to side viewed tank, foreground (class 0) and front viewed tank, foreground (class 1). The accuracy increased, ranging from 0.5786 (first nearest neighbor) to 0.6214 (seven nearest neighbors).

To summarize this portion of the feasibility study, classification based on Gabor coefficients is mediocre, at best.

These results should not, however, preclude further investigation. Preliminary results on the use of Gabor coefficients as classification features indicate promise. With the proper adjustments of the parameters, Gabor jet coefficients may be capable of accuracies well above those shown in this thesis.

Part 3. DESIGN OF AN OPTICAL GABOR TRANSFORM SYSTEM.

As an alternative method for generating the Gabor coefficients, an optical Gabor transform design is presented.

The design of the an optical Gabor transform system begins with a discussion of wavefront reconstruction and computer generated holograms.

Computer Generated Holograms. The concept of holograms came about in 1948 when Dennis Gabor proposed an imaging process called wave front reconstruction: interference between a coherent reference wave and the diffracted light from an object captured on photographic film, holography, followed by re-transmission of the image based on the interference fringes recorded on the film.

Coherent wave fronts necessitate the recording of amplitude and phase information about the waves. By interfering a known wavefront (reference beam) with an unknown wavefront (diffracted light from an object), the pattern captured on photographic film will be a function of the amplitude and phase of the unknown wavefront. Consider an unknown wavefront,

$$U(x,y) = U(x,y)\, e^{(-j\phi(x,y))} \qquad (11)$$

where $U(x,y)$ is the amplitude and $\phi(x,y)$ is the spatial phase of the unknown wavefront.

Let the reference wavefront be

$$R(x,y) = R(x,y)\, e^{(-j\delta(x,y))} \tag{12}$$

where $R(x,y)$ is the amplitude and $\delta(x,y)$ is the spatial phase of the reference wavefront.

When $R(x,y)$ interferes with $U(x,y)$, the resulting intensity pattern will be

$$I(x,y) = |R(x,y)|^2 + |U(x,y)|^2 + 2R(x,y)U(x,y)\cos(\delta(x,y) - \phi(x,y)) \tag{13}$$

where the first two terms are intensity dependent and the third term includes the difference between the spatial phases of the reference and unknown wavefronts.

Once $I(x,y)$, the hologram of the unknown object, has been recorded, the transparency of the hologram is illuminated by the reference beam to reconstruct the unknown wavefront. The resulting transmission wavefront will consist of four terms

$$R(x,y)\ T_f(x,y) = T_b R + \beta' U U^* R + \beta' R^* R U + \beta' R R U^* \qquad (14)$$
$$= A_1 + A_2 + A_3 + A_4$$

where $T_f(x,y)$ is the amplitude transmittance of the film which is comprised of a uniform "bias" transmittance, T_b, set by the reference and a "slope-exposure time" term, β' (scalar), taken from the t-E curve for the photographic film.

The second term, A_2, is the reference wavefront scaled by $\beta'|U(x,y)|^2$. The third term, A_3, is the unknown wavefront scaled by $\beta'(x,y)|R(x,y)|^2$. The fourth term is a combination of the unknown and reference wavefronts

$$A_4 = (R(x,y))^2\ e^{(-j2\delta(x,y))}\ U(x,y)\ e^{(+j\phi(x,y))} \qquad (15)$$

and is a function of both the reference and unknown wavefronts (8:198-273).

The third term is of interest since it possesses a scaled version of the unknown wavefront. Figure 4.12a/b depicts the two step, wavefront reconstruction process. In figure 4.12a, the interference between the unknown wavefront and the reference wavefront creates the fringe pattern recorded on film. In figure 4.12b, the film is illuminated by the reference beam to transmit the

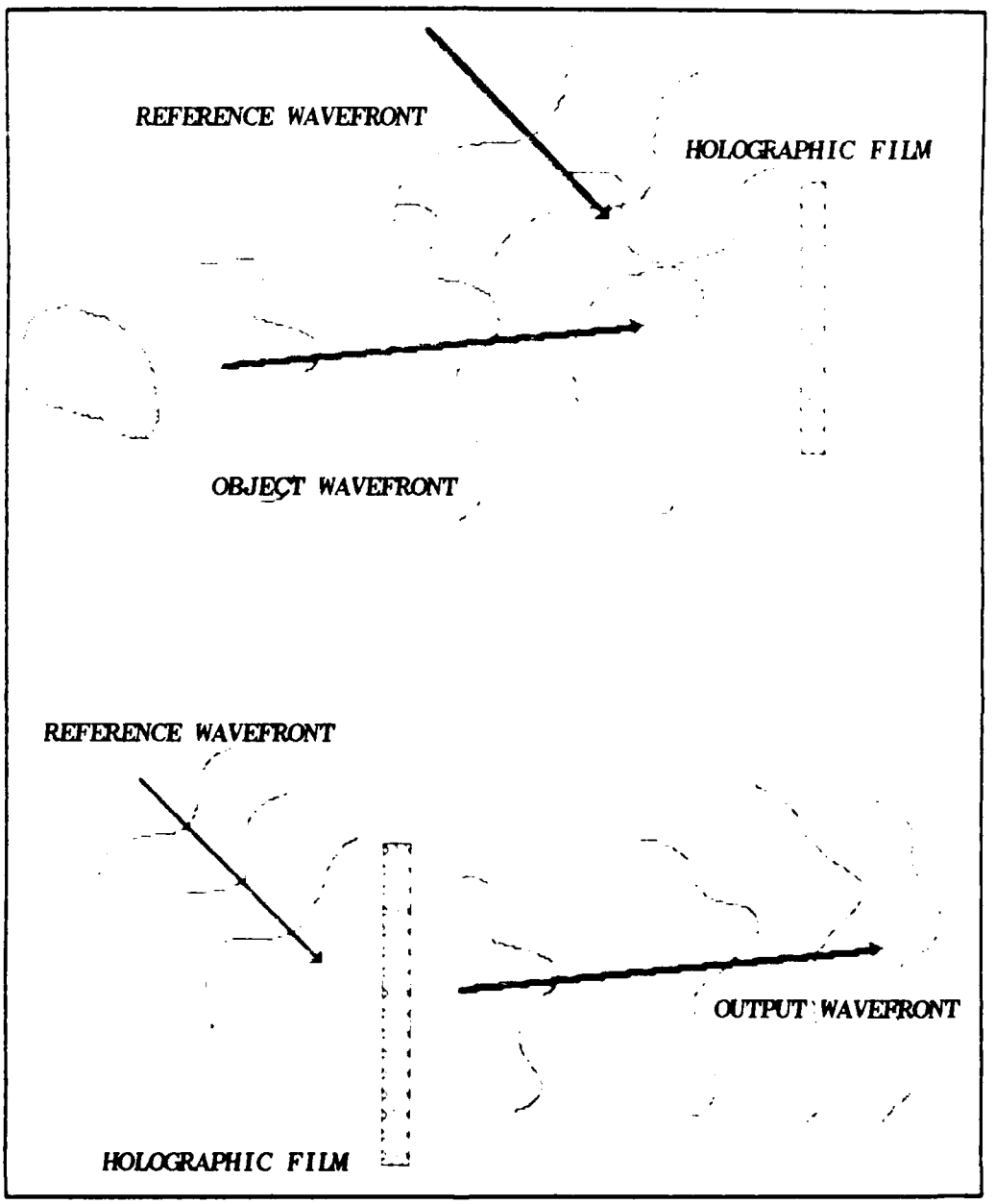


Figure 4.12. Wavefront reconstruction: Holography. (a) Creating the interferogram. (b) Reconstructing the hologram of the object via illumination.

unknown object beam. Note the incident angle of the reference wavefront relative to the object wavefront in both 13a and 13b.

The concept of wavefront reconstruction can be simulated using a computer generated hologram. The pattern resulting from the interference between a reference beam and an object of interest is comprised of magnitude and phase. Often, the magnitude and phase of individual points can be digitally calculated and plotted as a real, non-negative function using a plotting device. The resulting plotted image can then be photoreduced onto film to produce a transparency for use in an optical system (19:153-192)

The computer generated hologram for the cosine Gabor functions are degenerate in the sense that the magnitude of the function will produce a sufficient focal plane filter to perform optical Gabor transforms.

Figure 4.13 depicts the proposed design for optical Gabor transformation. A Helium Neon (He-Ne) laser beam is expanded (collimated) and impinges on a transparency of the target image which resides in the front focal plane of a Fourier transforming lens.

The target image is Fourier transformed. In the back focal plane of the Fourier transforming lens is a computer generated hologram of the complex conjugate of the Fourier transform of the Gabor function (see inset, Figure 4.13).

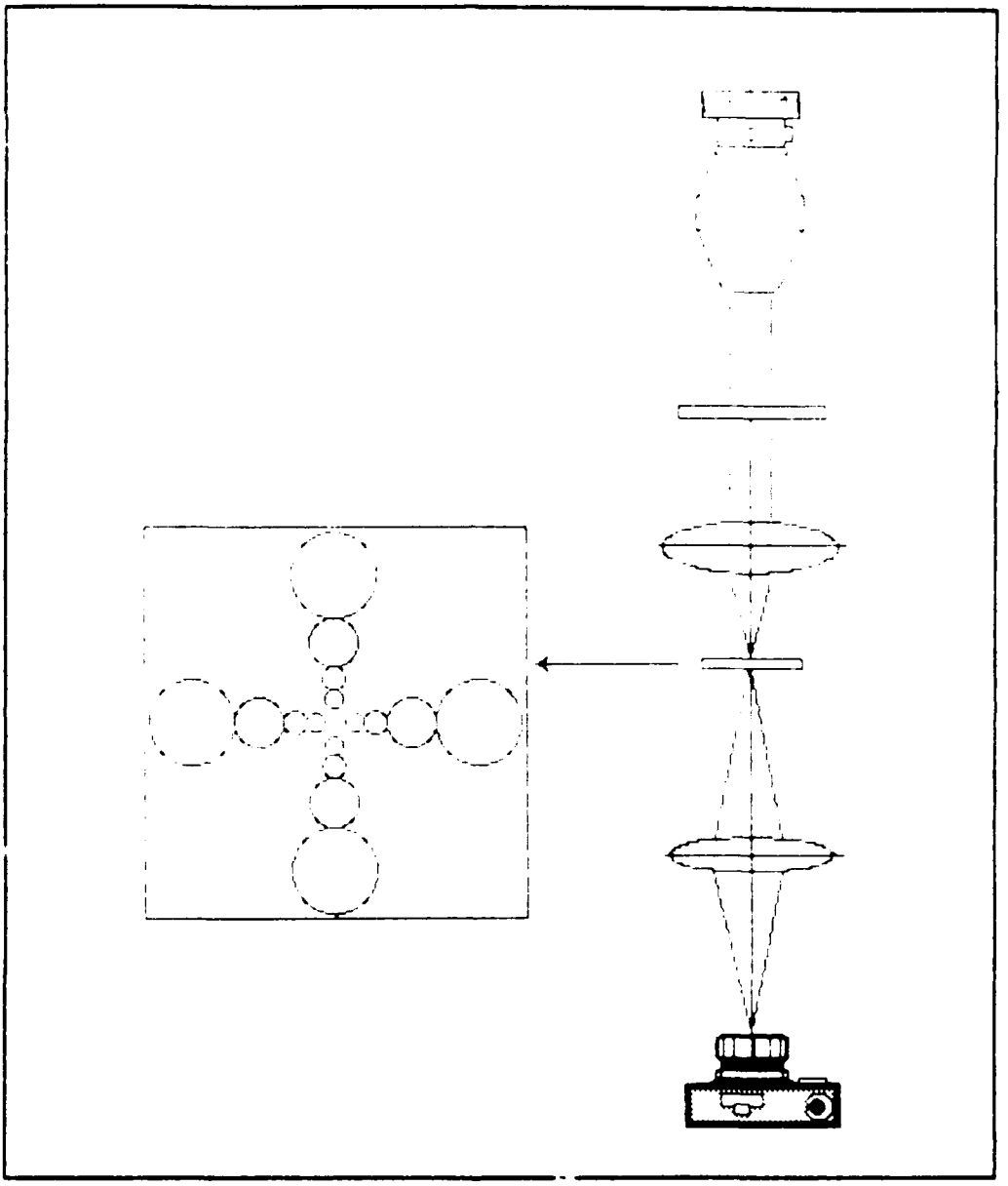


Figure 4.13. Schematic for proposed optical Gabor transform system. Collimator, Fourier transforming lens, Hologram plane, Fourier transforming lens, Camera.

A significant difference between the optical implementation and the digital implementation is that the optical implementation will superimpose images based on complex fields. The digital implementation superimposes images based on intensities. Therefore, if the optical implementation is to work, the fields must add coherently to provide segmentation.

One possible way of implementing the optical Gabor filter is to selectively introduce the Gabor orientations and frequencies such that only one set of Gaussian "circles" are present during each run. The resulting correlation image can be sensed by the CCD camera after the second Fourier lens.

A point for point multiplication will take place at the back focal plane of the Fourier transforming lens. Recall, point for point multiplication of the Fourier transform of the input image with the complex conjugate of the Fourier transform of the Gabor function results in correlation after inverse Fourier transformation.

The back focal plane product is Fourier transformed through another Fourier transforming lens and imaged onto a CCD camera. The signal processing from the CCD camera to the video display will flip the image about the x and y axes to properly display the image.

As each orientation and frequency is produced, the signal processing and storage of images can be accomplished. After the

desired number of orientations and frequencies are produced, superposition of the images can be accomplished. This will effectively be the optical implementation of the digital simulation presented in this thesis.

V. Conclusion and Recommendations

Conclusion.

The first, and foremost, task of any target recognition and classification system is to segment the target from the background. However, segmentation is not a trivial task.

One method for image segmentation discriminates based on frequency content and its corresponding distribution within an image. Gabor transforms use these image features to provide a means for discriminating regions of interest (regions of relatively high intensity) from less interesting regions (regions possessing less intensity) within FLIR images.

Gabor transforms use linear Fourier transform techniques to segment images, which significantly reduces the computational expense and complexity as opposed to those proposed by earlier researchers attempting FLIR image segmentation.

Gabor transforms do not require *a priori* information to segment effectively. Hence, a detection system using Gabor transforms should function in environments where information is not readily available, such as in the front end of a "smart" missile acquiring, tracking, and guiding to termination autonomously.

The dilating nature of the self similar and modified self similar Gabor functions is important from the standpoint of terminal

guidance. A missile may acquire a target several kilometers away. As such, the target may appear to be a small "blob". However, as the missile begins descending onto the target, the target size will increase relative to the missile's field of view. The segmentation scheme must take into account this change. Since Gabor functions can dilate (expand or contract) depending on the size of the target, they can easily accommodate dynamic changes in target size.

Gabor transforms are extremely sensitive to spatial frequencies and their respective orientations. For instance, frequency 3, orientations 45 and 135 degrees provide effective segmentation of some of the FLIR images whereas frequency 3, orientations 0 and 90 degrees provide effective segmentation for others. In some instances, frequency 4 provide more effective segmentation than frequency 3.

Consequently, constructing a filter which possesses several orientations and several frequencies, such as that proposed in the optical Gabor design, will be very specific in its spatial frequency and orientational correlation. The filter will ring, so to speak, within very specific regions, hence segmentation is possible.

Sine wave Gabor functions are edge detectors, whereas cosine Gabor functions tend to "fill in" regions of relatively slow change. Note, though, that both the sine wave and the cosine wave Gabor functions are subject to a global average intensity. This

dependency may reduce the Gabor function's effectiveness to discriminate low intensity targets. Therefore, "Lambertization" of the image may be required prior to Gabor transformation (See Appendix O for details on Lambertization).

By combining sine wave Gabor functions with cosine wave Gabor functions, complete segmentation, edge detection with body filling, of FLIR images is accomplished.

Gabor transforms have proven extremely useful for segmenting the FLIR images used in this research. The techniques used for classification produced promising results. Preliminary indications show that, with proper adjustment, the Gabor based classification system may produce increased classification accuracy(ies).

In summary, the results of this research effort show that Gabor transforms provide a fast, complete linear segmentation scheme and may very well become the pre-filtering standard for all image processing techniques.

Recommendations.

Digital Research.

Modified Self Similar Gabor Functions. Interesting future research using the digital implementation of Gabor transforms may include varying the number of cycles under the Gaussian envelope of the self similar Gabor function, yet retain its dilating nature. Including this feature using self similar Gabor functions renders them useful for segmentation, thereby combining the "best" features of both the self similar and non-self similar Gabor functions. Results indicate that combining the non-self similar features with self similar features result in an excellent segmenter (See Appendix M).

Multi-Sensor Fusion Analog. Another area of interest may include developing digital simulations of a multi-sensor fusion Gabor segmenter based on FLIR and Laser Designated Range (LADAR) information in an effort to improve the efficiency and accuracy of detection and classification.

Vector Space Warping for Classification. Vector space warping is an effort to make the jet vector approach for classification scale invariant.

The idea is patterned after a speech recognition scheme called time warping. Speech signals of the same utterance are variable in duration. Hence, to correctly identify specific speech utterances,

the feature used for recognition must be variable. A difficulty with time warping is temporal alignment of the speech signals. Once aligned, the test vectors (of length N) are compared to the template vector (of length M). Simplistically speaking, the vector comparison producing the minimum distance trajectory along the diagonal (or closest to the diagonal) of the N by M matrix should correspond to the correct word. For a more in-depth discussion of dynamic time warping refer to Rabiner, *et. al.* (17:575-582).

Similarly, same objects can have different sizes, depending on range of view, and the metric used for determining what these objects are should account for this change.

Recall, the jet vector approach adopted for this research used eight components, four orientations at a Gabor window size of 16 by 16 and four orientations at a Gabor window size of 8 by 8, irrespective of the target size (range).

Targets further down range from the FLIR detector platform could not be reliably identified using Gabor correlation coefficients, although they could readily be segmented using Gabor transforms. This was due, in part, to the fact that over fifty percent of the pixels within the 16 by 16 window sizes corresponded to surroundings and not the target.

Hence, only four coefficients should be used to identify the targets further down range. The question is raised, " How can we

compare down range tank vectors to up range tank vectors ? ". A means for comparing down range target vectors to up range target vectors should be developed. One possible means of accomplishing this task is to employ vector space warping, comparing four dimensional vectors to eight dimensional vectors using a similar technique as time warping.

Neural Networks. Neural networks are devices which employ densely interconnected, simple computational units called neurons. These neurons are usually connected to form distinct layers, such as can be found in a multilayer perceptron network (14:1-136).

Multilayer Perceptron Approach. The multilayer perceptron has an input layer, two or more inner or hidden layers, and an output layer. Each node within a layer is connected to nodes in the previous layer. The connections are weighted, initially, in a random manner with low values (between 0.1 and 0.5). The network is given a set of inputs on which to train.

The output is compared with the input and the weights are changed. This process is continued until the network "settles" on an answer.

The input to the multilayer perceptron can be the eight vector components derived from the jets. Each of the eight input nodes can be connected to ten nodes in the first hidden layer. Each of the first hidden layer nodes, in turn, can be connected to ten nodes

in a second hidden layer. Each node in the second hidden layer can be connected to an output layer containing five nodes (each node corresponding to one of the five locations determined by the eight dimensional jet vector approach).

Optical Research.

Non-linear Crystals. The use of non-linear crystal media such as Lithium Niobate or Barium titanate may replace the computer generated hologram filter approach to allow for near real-time holography. For more information about non-linear crystals, two text books are excellent references (24:1-156, 25:1-589).

REFERENCES

1. Bastiaans M., "Gabor's Expansion of a Signal into Gaussian Elementary Signals.", Proc. IEEE, vol. 68, pp. 538 - 539, 1980.

2. Bush, Lawrence F., "The Design of An Optimum Alphanumeric Symbol Set for Cockpit Displays.", Master's Thesis, Air Force Institute of Technology, pp. 1 - 96, 1977.

3. Cooley, J. W., and J. W. Tukey, "An Algorithm for Machine Calculation of Complex Fourier Series.", Mathematics of Computation., vol. 19, pp. 297 - 301, April 1965.

4. Daugman, John G., "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression.", IEEE Transactions on Acoustics, Speech, and Signal Processing., vol. 36, pp. 1169 - 1179, July 1988.

5. Flaton, Kenneth A., and S. T. Toborg, "An Approach to Images Recognition Using Sparse Filter Graphs.", Proceedings from IJCNN, Vol. 1, pp. I3i3 - I320, June 1989.

6. Gabor, Dennis, "Theory of Communication.", J. Inst. Elec. Eng., vol. 93, pp. 429 - 457, 1946.

7. Gonzalez, R., and P. Wintz, Digital Image Processing., Second Edition, Addison-Wesley, Mass., pp. 105-109, 1987.

8. Goodman, Joseph W., Introduction to Fourier Optics., McGraw - Hill Book Company, New York, pp. 1 - 287, 1968.

9. Grossberg, Stephen, "Cortical Dynamics of Three Dimensional Form, Color, and Brightness Perception: I. Monocular Theory.", Perception and Psychophysics., vol. 41 (2), pp. 87 - 116, 1987.

10. Grossmann, A., and J. Morlet, "Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape.", *SIAM J. Math. Anal.*, vol. 15, pp. 723 - 736, 1984.

11. Horev, Moshe, "Picture Correlation Model for Automatic Machine Recognition.", Master's Thesis, Air Force Institute of Technology, 1980.

12. Jones, Judson P., and L. A. Palmer, "An Evaluation of the Two Dimensional Gabor Filter Model of Simple Receptive Fields in Cat Striate Cortex.", *Journal of Neurophysiology.*, vol 58, no. 6, pp. 1233 - 1258, December 1987.

13. Kabrisky, Matthew, and Steve Rogers, Notes from Pattern Recognition I., EE 620, Air Force Institute of Technology, pp. 1 - 143, Winter Quarter 1989.

14. -----, Notes Based on Lectures about Biological and Artificial Neural Networks for Pattern Recognition, Air Force Institute of Technology, pp. 1 - 136, Winter Quarter 1989.

15. Lambert, L. C., "Evaluation and Enhancement of the AFIT Autonomous Face Recognition Machine.", Master's Thesis, Air Force Institute of Technology, pp. 1 - 86, 1987.

16. Mallat, Stephane, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation.", *IEEE: Transactions on Pattern Analysis and Machine Intelligence*, vol. II, no. 7, pp. 674 - 693, July 1988.

17. Rabiner, L. R., A. E. Rosenberg, and S. E. Levinson, "Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition.", *IEEE. Trans. on Acoustics, Speech, and Signal Processing.*, vol. ASSP-26, no. 6, December 1978.

18. Roberts, Richard E., M. Kabrisky, S. Rogers, and E. Amburn, "Three Dimensional Scene Analysis Using Stereo Based Imaging.", *SPIE: Applications of Artificial Intelligence VI.*, vol. 937, pp. 280 - 286, 1988.

19. Rogers, S. K., Notes for Advanced Topics in Optical Information Processing, EENG 715, Air Force Institute of Technology, pp. 153 - 192, Fall Quarter 1989.

20. Roggemann, Michael C., J. P. Mills, S. K. Rogers, and M. Kabrisky, "Segmentation of Noisy Range Images Using the Small Scale Planarity of Man-Made Vehicles.", submitted for publication in *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 18 pages, 1989.

21. Rudin, Walter, *Functional Analysis*, Mcgraw - Hill Book Company, New York, pp. 1 - 397, 1973.

22. Siebert, Michael, and A. M. Waxman, "Spreading Activation Layers, Visual Saccades, and Invariant Representations of Neural Pattern Recognition Systems.", *Neural Networks.*, vol. 2, pp. 9 - 27, 1989.

23. Turner, M. R., "Texture Discrimination by Gabor Functions.", *Biological Cybernetics.*, vol. 55, pp. 71 - 82, 1986.

24. Wood, E. A., *Crystals and Light. An Introduction to Optical Crystallography.*, Second Revised Edition, Dover Pub., Inc., New York, pp. 1 - 156, 1977.

25. Yariv A., and P. Yeh, *Optical Waves in Crystals. Propagation and Control of Laser Radiation.*, Wiley Interscience, John Wiley and Sons, New York, pp. 1 - 589, 1984.

Appendix A. Glossary of terms

1. Computer generated holograms - computer renderings of objects which, when illuminated by coherent beams (laser beams), generate a complete "three-dimensional" image of the object. It should be noted that Dennis Gabor is famous for being one of the "founding fathers" of holography.

2. Digital images - an image whose intensity is divided into a discrete valued array of elements which, when arranged properly, reconstruct the image.

3. Fast Fourier Transform (FFT) - the decomposition of a discrete Fourier transform into smaller transforms then recombining them to produce the complete Fourier transform of the image. Commonly called decimation in frequency or decimation in time transforms popularized by Cooley and Tukey (3:297-301).

4. Forward Looking InfraRed (FLIR) - a technique for detecting targets by the thermal gradient between the target and its surroundings.

5. Fourier Transform - a method to convert a space (time) signal to a spatial (temporal) frequency signal without loss of signal integrity.

6. Jet Vectors - A vector comprised of Gabor correlation coefficients associated with specific Gabor function frequencies and orientations.

7. Laser Designated Radar (LADAR) - a device that determines the range between an object and the platform on which the LADAR resides by illuminating the object with a laser beam.

8. Neural network - a densely interconnected set of simple, computational elements called nodes.

9. Segmentation - distinguishing targets from background based on a given feature such as temperature, range, planarity, or frequency content.

10. Self (Non-Self) Similarity - A characteristic of Gabor functions whereby the frequency of the sinusoidal cofactor is (is not) a direct function of or equivalent to the standard deviation of the Gaussian cofactor. For Non-Self similarity, the Gaussian standard deviation is fixed and the sinusoidal cofactor frequency is allowed to vary.

Appendix B. Speech signal processing using MathCAD.

SYNOPSIS: MathCAD template for calculating the correlation between the utterance "ONE", as spoken by Dr. Matthew Kabrisky, and the self similar Gabor function.

All input data are entered into the MathCAD template using the built in function READ. The Fourier transform of the Gabor function and the voice track data is calculated using the built in function CFFT, Complex Fast Fourier Transform. The Fourier transform of the voice data is multiplied by the complex conjugate of the Fourier transform of the Gabor function. The product of the point for point multiplication of the Fourier transform of the voice data by the complex conjugate of the Fourier transform of the Gabor function is inverse Fourier transformed using the built in function ICFFT, Inverse Complex Fast Fourier Transform. The results are displayed using built in graphics routines.

INPUT: Data from voice utterance sampled at 16 KHz.
Gabor function data, Filename: GBR1.DAT.

OUTPUT: The correlation between the speech utterance and the Gabor function.

One Dimensional Gabor Transforms. Speech Analysis.

The Zenith model 248 personal computer (PC) was used to generate the one dimensional Gabor transform analysis of the speech signal, "ONE", as spoken by Dr. M. Kabrisky. The primary mathematical program to analyze the utterance was MathCAD, purchased from Mathsoft, Inc., which performed complex Fourier transforms, complex inverse Fourier transforms, indexed data, and generated graphs for display.

The Gabor function was generated for a data block size of 512. The Gabor function was Fourier transformed using the complex Fast Fourier Transform (CFFT) algorithm provided by the MathCAD software. The transform of the Gabor function was converted to its complex conjugate.

The word utterance "ONE" was sampled at 16 kilohertz (KHz) then divided into blocks of 512 data points (approximately 32 millisecond windows). The CFFT routine was performed on each block of 512 utterance sample points. Then a point for point multiplication of the CFFT of the utterance with the complex conjugate of the CFFT of the Gabor function was accomplished. After inverse complex Fast Fourier transforming the product, the result was the correlation between the word utterance and the Gabor function.

Index counter:

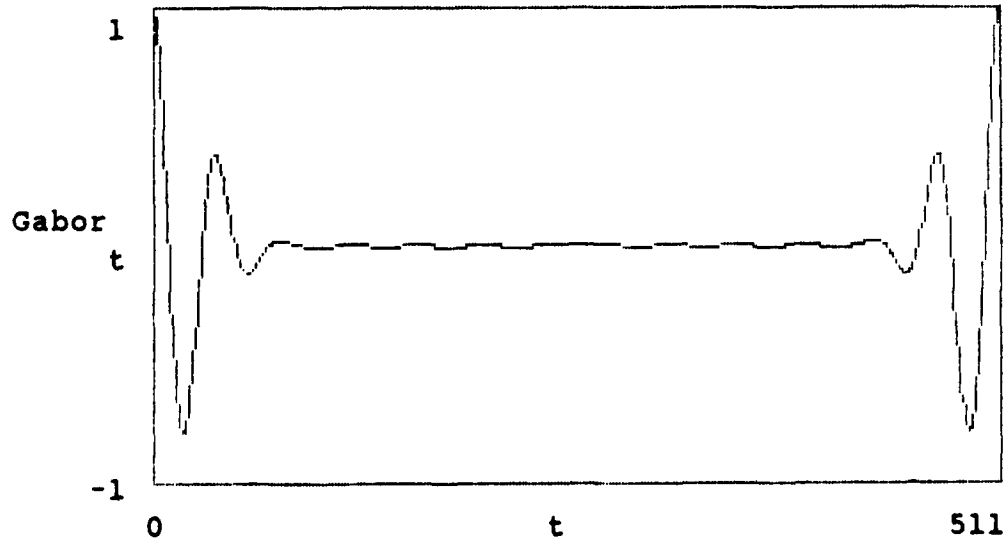
$$t := 0 \,..511$$

Read the speech utterance: v1 corresponds to the data points from 1024 to 1535. v2 corresponds to the data points from 1536 to 2047.

$$v1_t := READ(one1k\ kab) \qquad v2_t := READ(one12\ kab)$$

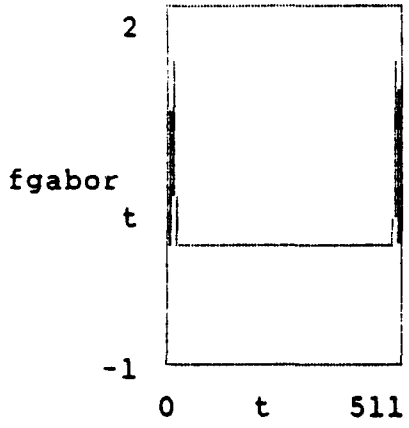
Read the Gabor Function:

$$Gabor_t := READ(gbr1\ dat)$$

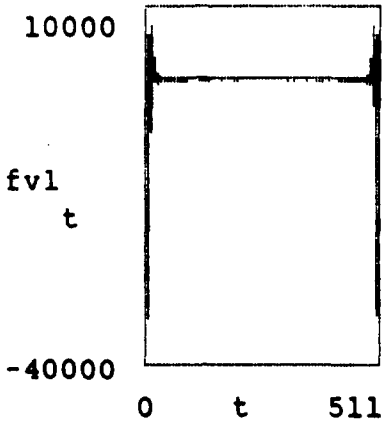


Fourier transform of Gabor and voice tracks

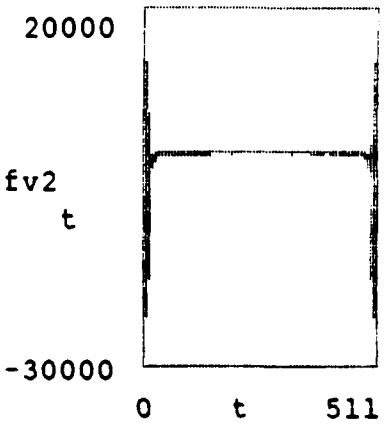
fgabor := cfft(Gabor)



fv1 := cfft(v1)



fv2 := cfft(v2)

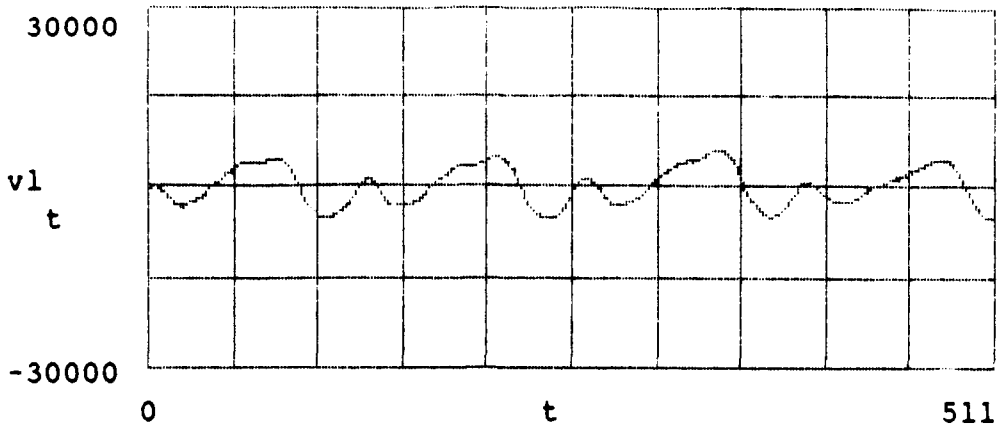


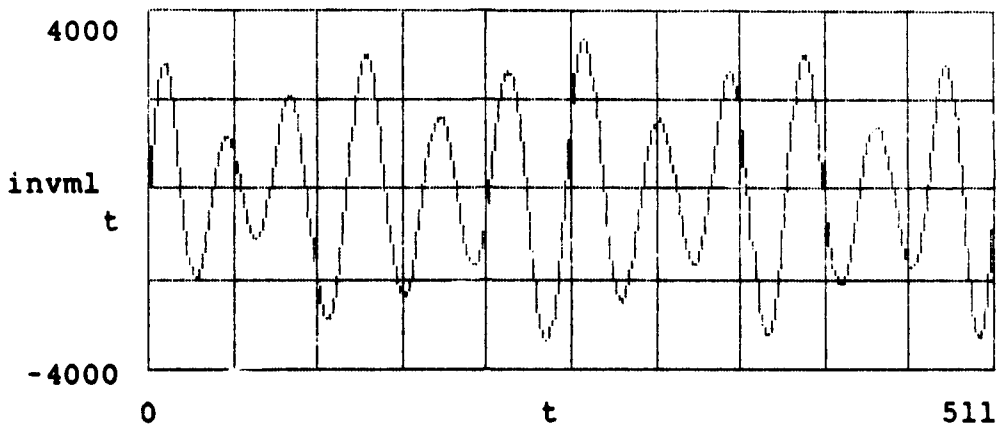
Multiply the Fourier transform of v's with the complex conjugate of the Fourier transform of Gabor.

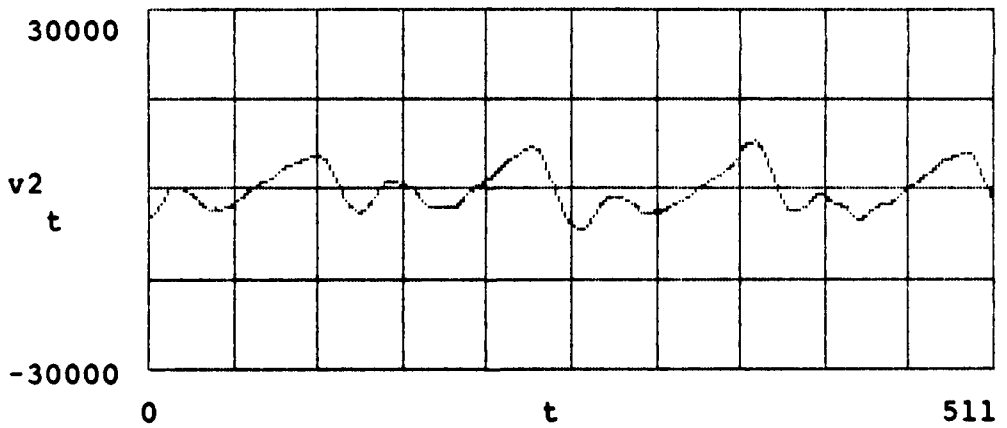
$$m1_t := fv1_t \cdot \overline{fgabor_t} \qquad m2_t := fv2_t \cdot \overline{fgabor_t}$$

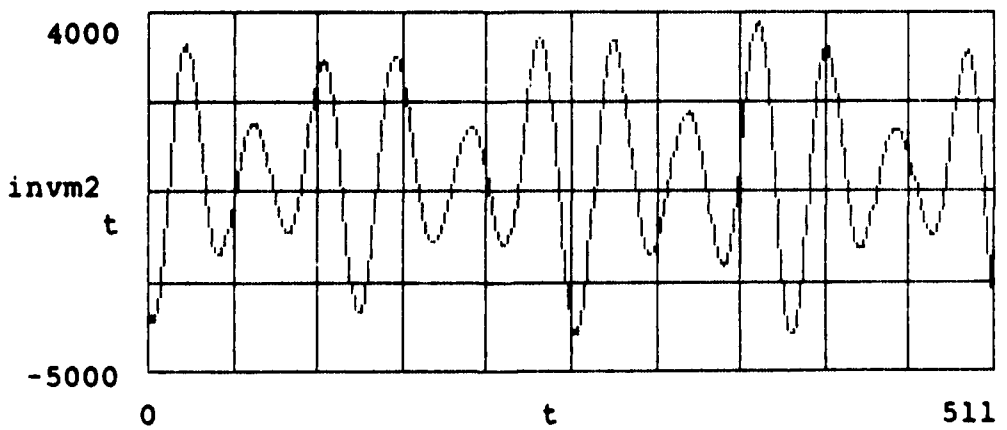
Inverse Fourier transform of the m functions

invm1 := icfft(m1) invm2 := icfft(m2)


30000
v1
t
-30000
0
t
511


4000
invm1
t
-4000
0
t
511


30000
v2
t
-30000
0
t
511


4000
invm2
t
-5000
0
t
511

Appendix C. Part 1. Sine wave Gabor transform FORTRAN source code.

BSG.FOR

Appendix C. Part 2. Cosine wave Gabor transform FORTRAN source code.

BCG.FOR

SYNOPSIS: Part 1 / Part 2. The BSG.FOR code reads in image data stored as two's complement BYTE format and converts it to integer format. After reading in the data, the program generates a filename for the Gabor filtered image. Depending on the user's choice, the program then generates a self similar, modified self similar, or a non-self similar Gabor function on center within a 256 by 512 pixel block. Several parameters are passed to the Gabor function generating subroutine. They are the window size, the frequency of the sinusoidal variation cofactor (initially set to the lowest frequency of interest then incremented according to the user's preference), the standard deviation at the N/2 point, current in-plane rotation (orientation of the sinusoidal variation), and the user's choice of sine wave or cosine wave variation (the only difference between part 1 and part 2 is this choice). The program then takes the Fourier transform of the image and of the Gabor function, performs a point for point multiplication of the Fourier transform of the image with the complex conjugate of the Fourier transform of the Gabor function, then inverse Fourier transforms the product. The program then sends the result to the aforementioned output file. The program continues until all images of the correlation between the Gabor functions possessing the frequencies and orientations of interest and the raw FLIR image are generated.

For the readers' information: The MOD command, found in most versions of FORTRAN, takes the remainder of the quotient of image value and 256, then subtracts 128 from the remainder. Values in excess of 127 are "folded over", ie. 128 is equivalent to zero, using the MOD command. This operation ensures the value of the BYTE integer to be within the prescribed range. It should be noted that the BYTE option for integer representation is specific to VMS FORTRAN. However, the LOGICAL*1 integer format can be found in most versions of FORTRAN. If the reader desires to use the existing code under other operating systems, consult the FORTRAN manual specific to the operating system to see if BYTE format is a viable option. If not, the LOGICAL*1 format should work.

INPUT: Raw FLIR image, 240 by 640 pixels

OUTPUT: Gabor filtered images.

```
      PROGRAM GB_MAIN
      DIMENSION       NAR(256,512),NMAG(256,512)
      DIMENSION       GBR(256,512),xar(256,512)
      CHARACTER*24    NAME,NAMEOUT,lml
      integer         pass



      type*,'************************************************************'
      type*,'*  Welcome to the Gabor transform program. This program *'
      type*,'*  will generate the Gabor transform of any 240 by 640, *'
      type*,'*  byte format image. If you desire help before the pro-*'
      type*,'*  gram, begins execution, ie. before the last prompt,  *'
      type*,'*  please type 911 at the prompt.  The help screen will *'
      type*,'*  be invoked, and you will be able examine any para-   *'
      type*,'*  meter for its function.                              *'
      type*,'*  Remember, after the program begins execution, you    *'
      type*,'*  will not be able to see the help files. Also, you    *'
      type*,'*  cannot get into the help routine at the FLIR name    *'
      type*,'*  prompt.                                             *'
      type*,'*  This help file is not extensive but gives a brief    *'
      type*,'*  description for each parameter the user inputs.      *'
      type*,'************************************************************'
      type*,' '
      type*,'                                --- Kevin W.Ayer, CAPT, USAF'
      type*,'                                    3 September 1989'
      TYPE*,' '
      type*,'       Do you wish to see the help files now?'
      type*,'           Enter a 911 if you so desire.'
      read(5,*)nhelp
      if(nhelp.eq.911) then
      call help
      endif

 1    CONTINUE


C-------------------------------------------------C
C  THE FOLLOWING PARAMETER (NAME) SETS THE NAME   C
C  OF THE INPUT FLIR IMAGE FILE.                  C
C-------------------------------------------------C

      TYPE*,'NAME OF FLIR IMAGE FOR GABOR TRANSFORMING'
      ACCEPT 30,NAME

C-------------------------------------------------C
C  THE FOLLOWING PARAMETERS SET THE NUMBER OF     C
C  PASSES THE PROGRAM TAKES RELATIVE TO THE       C
C  FREQUENCY(IES) OF INTEREST.                    C
C  THEY ARE ALSO THE FREQUENCY(IES)ASSOCIATED     C
C  WITH THE NON-SELF SIMILAR GABOR FUNCTION(S).   C
C-------------------------------------------------C

      TYPE*,'LOWEST FREQUENCY?'
      READ(5,*)NLOF
      if(nlof.eq.911) then
      call help
      endif
      TYPE*,'HIGHEST FREQUENCY?'
      READ(5,*)NHIF
```

```
        ir(nhif.eq.911) then
        call help
        endif

C-------------------------------------------------C
C  THE FOLLOWING PARAMETER SETS THE STD. DEV. OF  C
C  THE SELF SIMILAR GABOR FUNCTION(S).            C
C-------------------------------------------------C

        TYPE*,'STANDARD DEVIATION AT THE MAX EXTENT?'
        READ(5,*)NLOF1
        if(nlof1.eq.911) then
        call help
        endif

C-------------------------------------------------C
C  THE FOLLOWING PARAMETERS ESTABLISH THE MAXIMUM C
C  ROTATION (ORIENTATION) AND THE INCREMENT OF    C
C  ROTATION FOR THE GABOR FUNCTIONS (NSS AND SS)  C
C-------------------------------------------------C

        TYPE*,'MAXIMUM ANGULAR ORIENTATION (IN DEGREES)?'
        READ(5,*)MAX_ROT
        if(max_rot.eq.911) then
        call help
        endif

        TYPE*,'INCREMENT OF ANGULAR ORIENTATION (IN DEGREES)?'
        READ(5,*)INC_ROT
        if(inc_rot.eq.911) then
        call help
        endif

C-------------------------------------------------C
C  THE FOLLOWING PARAMETER SETS THE SIZE OF THE   C
C  GABOR 'PATCH'.                                 C
C-------------------------------------------------C

        TYPE*,'SIZE OF THE GABOR WINDOW? MINIMUM=8, MAXIMUM=128'
        READ(5,*)NSIZE
        if(nsize.eq.911) then
        call help
        endif

C-------------------------------------------------C
C  THE FOLLOWING PARAMETERS SET THE TYPE OF GABOR C
C  FUNCTION USED FOR IMAGE PROCESSING.  NCH SETS  C
C  THE SELF SIMILAR OR NON-SELF SIMILAR (SS=0,    C
C  NSS=1) AND NSIN_NCOS SETS THE SINE OR COSINE   C
C  VARIATION.                                     C
C-------------------------------------------------C

        TYPE*,'REGULAR SELF SIMILAR=0, MODIFIED SELF SIMILAR=1'
        TYPE*,'NON-SELF SIMILAR=2'
        READ(5,*)NCH
        if(nch.eq.911) then
        call help
        endif

        type*,'SEGMENTATION=0, CLASSIFICATION=1'
```

```
        READ(5,*)NCHSC
        if(nchsc.eq.911) then
        call help
        endif

        TYPE*,'SINE=0, COSINE=1'
        READ(5,*)NSIN_NCOS
        if(nsin_ncos.eq.911) then
        call help
        endif

C-------------------------------------------------C
C  NROT IS THE INITIAL ROTATION (ORIENTATION).    C
C  NDIFF1 IS THE NUMBER OF PASSES RELATIVE TO THE C
C  THE FREQUENCY(IES) OF INTEREST.  MADIP1 IS THE C
C  NUMBER OF PASSES RELATIVE TO THE NUMBER OF     C
C  ORIENTATIONS OF INTEREST SUCH THAT THE TOTAL   C
C  NUMBER OF PASSES IS NDIFF1*MADIP1.             C
C-------------------------------------------------C

        NROT=0
        NDIFFP1 = NHIF - NLOF + 1
        MADIP1 = MAX_ROT/INC_ROT +1

        IF(NCHSC.EQ.0) THEN
           NDIFFP1=2
        ENDIF

        NFLG_FILE = 0
        NPASS = 1


C--   THIS CALL STATEMENT EXECUTES A SUBROUTINE WHICH        --C
C--   RETRIEVES A BYTE FORMATTED IMAGE FROM MEMORY.          --C
C--   LM1 IS THE FILENAME FOR THE LAMBERTIZED FLIER IMAGE    --C

        lm1='lambert1.flr'

        CALL GET_FLIR_IM(NAME,NAR,240,640)

C--   THIS CALL STATEMENT PUTS THE LAMBERTIZED VERSION OF    --C
C--   IMAGE INTO FILNAME LM1.                                --C

        CALL PUT_FLIR_IM(lm1,NAR,240,640)

C--   THIS PORTION OF THE MAIN PROGRAM BEGINS THE SEQUENCE   --C
C--   OF OPERATIONS NECESSARY TO GENERATE THE GABOR COR-     --C
C--   RELATION(S) OF A RAW FLIR IMAGE.                       --C


        DO 10 I=1,NDIFFP1

           DO 20 J=1,MADIP1


C--   THIS CALL STATEMENT EXECUTES A SUBROUTINE WHICH        --C
C--   ESTABLISHES THE NAME OF THE GABOR FILTERED IMAGE       --C

                CALL OUT_NAME(NFLG_FILE,NAMEOUT)
```

```
C--   THESE CALL STATEMENTS EXECUTE SUBROUTINES WHICH        --C
C--   GENERATE EITHER SELF SIMILAR (SS), MODIFIED SELF       --C
C--   SIMILAR(MSS), OR NON-SELF SIMILAR (NSS) GABOR          --C
C--   FUNCTION(S).                                           --C

                IF(NCH.EQ.0) THEN
                 CALL SS_GABOR(GBR,NSIZE,NLOF1,NROT,NSIN_NCOS)
                ELSEIF(NCH.EQ.1) THEN
                 CALL MSS_GABOR(GBR,NSIZE,NLOF1,NLOF,NROT,NSIN_NCOS)
                ELSE
                 CALL NSS_GABOR(GBR,NSIZE,NLOF,NROT,NSIN_NCOS)
                ENDIF

C--   THIS CALL STATEMENT EXECUTES THE GABOR TRANSFORMATION --C
C--   SUBROUTINE WHICH USES A 2-D FFT SUBROUTINE.           --C

               pass=1

                CALL GABOR_TRANSFORM(GBR,NAR,NMAG,256,512,pass)


C--   THIS CALL STATEMENT EXECUTES A SUBROUTINE WHICH SENDS --C
C--   THE GABOR FILTERED IMAGE TO MEMORY.                   --C


      CALL PUT_FLIR_IM(NAMEOUT,NMAG,240,640)



                NFLG_FILE = NFLG_FILE + 1

C--   THE ROTATION IS INCREASED BY THE INCREMENT AND THE    --C
C--   NUMBER OF THE PASS IS INCREMENTED BY 1                --C

                NROT=NROT+INC_ROT
                NPASS = NPASS + 1

 20       CONTINUE

C--   CHANGE THE WINDOW SIZE IF GENERATING SELF SIMILAR     --C
C--   GABOR FUNCTION(S).                                    --C

      IF(NCHSC.EQ.0) THEN
         NSIZE=NSIZE/2
      END IF

C--   RESET THE NROT PARAMETER TO ZERO DEGREES.             --C

         NROT = 0

 10    CONTINUE


C--   REQUEST USER TO EXPRESS DESIRE FOR CONTINUING THE     --C
C--   PROGRAM.                                              --C

      TYPE*,'DO YOU WISH TO CONTINUE? NO=0, YES=1'
      READ(5,*)NBO
      IF(NBO.EQ.1) THEN
          GOTO 1
```

```
        END IF

 30     FORMAT(A24)
        END



        SUBROUTINE GET_FLIR_IM(NAME,AR,ROW_NUM,COL_NUM)

        IMPLICIT          INTEGER (A-Z)
        BYTE              AR_BYTE(240,640)
        DIMENSION         AR(256,512)
        REAL              TOP(512),BOTTOM(512),THE_MAX,XAR(256,512)
        REAL              overall_avg,avg(256,512),XTot,the_min
        real              x1,x2,x3,x4,XTOP,XBOTTOM,RTOP,RBOTTOM
        REAL              RXAR(256,512)
        CHARACTER*24      NAME

C--  OPENS THE LOGICAL UNIT 10 FOR READING INFORMATION FROM  --C

        OPEN(10,FILE=NAME,STATUS='OLD',FORM='UNFORMATTED',
     +  RECORDTYPE='VARIABLE',ERR=20)
 20     OPEN(10,FILE=NAME,STATUS='OLD',FORM='UNFORMATTED',
     +  RECORDTYPE='FIXED')


C--                                                      --C
C--  ZEROING OUT THE TRANSFER DATA STRUCTURES            --C
C--                                                      --C

        DO R=1,256
           DO C=1,512
                XAR(R,C)=0.
           END DO
        END DO


C--                                                      --C
C--  READING IN THE BYTE FORMATTED DATA                  --C
C--                                                      --C

        DO 100 R=1,ROW_NUM
           READ(10) (AR_BYTE(R,C),C=1,COL_NUM)
 100    CONTINUE

C--                                                      --C
C--  PUTTING THE DATA INTO A 256 BY 512 FORMAT           --C
C--                                                      --C

        RP = 9
        DO R=1,240
           DO C=65,576
              CP=C-64
              IF(AR_BYTE(R,C).LT.0) THEN
                AR(RP,CP)=AR_BYTE(R,C)+256
              ELSE
                AR(RP,CP)=AR_BYTE(R,C)
              ENDIF
           END DO
```

```
         RP=RP+1
         END DO


C--   CONVERTING THE INTEGER FORMAT TO REAL FORMAT          --C

         DO R=1,256
            DO C=1,512
               XAR(R,C)=REAL(AR(R,C))
            END DO
         END DO

         xtot=0.


C--   FINDING THE HIGHEST AND LOWEST PIXEL INTENSITY VALUES  --C

         DO C=1,512
            XTOP=0.
            XBOTTOM=1000.
            DO R=1,256
               X2=XAR(R,C)
               RTOP=XTOP-X2
               RBOTTOM=XBOTTOM-X2
               IF(RTOP.LT.0) THEN
                  XTOP=X2
               ENDIF
               IF(RBOTTOM.GT.0) THEN
                  XBOTTOM=X2
               ENDIF
            END DO
            TOP(C)=XTOP
            BOTTOM(C)=XBOTTOM
         END DO



         THE_MAX=0.
         THE_MIN=1000.

         DO C=1,512

            X3=TOP(C)
            X4=BOTTOM(C)
            XTOP=THE_MAX-X3
            XBOTTOM=THE_MIN-X4
            IF(XTOP.LT.0) THEN
               THE_MAX=X3
            ENDIF
            IF(XBOTTOM.GT.0) THEN
               THE_MIN=X4
            ENDIF
         END DO


         IF(THE_MAX.EQ.THE_MIN) THEN
            THE_MAX=THE_MAX+1.
         ENDIF
```

```
C--   LAMBERTIZATION OF INPUT FLIR IMAGE                          --C

C--   REQUEST USER TO SPECIFY IF LAMBERTIZATION IS DESIRED        --C

      type*,'DO YOU WISH TO LAMBERTIZE THE IMAGE? NO=0, YES=1'
      READ(5,*)LCH
      IF(LCH.EQ.1) THEN

      do i=2,255
         im1=i-1
         ip1=i+1
         do j=2,511
            jm1=j-1
            jp1=j+1
            avg(i,j)=(xar(im1,jm1)+xar(im1,j)+xar(im1,jp1)+
     +                xar(i,jm1)+xar(i,jp1)+xar(ip1,jm1)+
     +                xar(ip1,j)+xar(ip1,jp1)+XAR(I,J))/9.

         end do
      end do

      DO R=1,256
         DO C=1,512
            RXAR(R,C)=127.+(XAR(R,C)-AVG(R,C))
         END DO
      END DO

      ENDIF


C--   SCALING THE PIXEL INTENSITY VALUES BETWEEN 0 AND 255        --C

      DO R=1,256
         DO C=1,512
            RXAR(R,C)=255.*((RXAR(R,C)-THE_MIN)/(THE_MAX-THE_MIN))
            AR(R,C)=IFIX(RXAR(R,C))
         END DO
      END DO

      CLOSE(10)
      RETURN
      END




      SUBROUTINE OUT_NAME(NFLG,NAMEOUT)

      CHARACTER*24     NAMEOUT


      IF(NFLG.EQ.0) THEN
         NAMEOUT='RF1:SF1R0_RJ.FLR'
      ELSEIF(NFLG.EQ.1) THEN
```

```
   NAMEOUT='RF1:SF1R45_RJ.FLR'
ELSEIF(NFLG.EQ.2) THEN
   NAMEOUT='RF1:SF1R90_RJ.FLR'
ELSEIF(NFLG.EQ.3) THEN
   NAMEOUT='RF1:SF1R135_RJ.FLR'
ELSEIF(NFLG.EQ.4) THEN
   NAMEOUT='RF1:SF1R0_RJ.FLR'
ELSEIF(NFLG.EQ.5) THEN
   NAMEOUT='RF1:SF1R45_RJ.FLR'
ELSEIF(NFLG.EQ.6) THEN
   NAMEOUT='RF1:SF1R90_RJ.FLR'
ELSEIF(NFLG.EQ.7) THEN
   NAMEOUT='RF1:SF1R135_RJ.FLR'
ELSEIF(NFLG.EQ.8) THEN
   NAMEOUT='RF1:SF1R0_RJ.FLR'
ELSEIF(NFLG.eQ.9) THEN
   NAMEOUT='RF1:SF1R45_RJ.FLR'
ELSEIF(NFLG.EQ.10) THEN
   NAMEOUT='RF1:SF1R90_RJ.FLR'
ELSEIF(NFLG.EQ.11) THEN
   NAMEOUT='RF1:SF1R135_RJ.FLR'
ELSEIF(NFLG.EQ.12) THEN
   NAMEOUT='RF1:SF1R0_RJ.FLR'
ELSEIF(NFLG.EQ.13) THEN
   NAMEOUT='RF1:SF1R45_RJ.FLR'
ELSEIF(NFLG.EQ.14) THEN
   NAMEOUT='RF1:SF1R90_RJ.FLR'
ELSEIF(NFLG.EQ.15) THEN
   NAMEOUT='RF1:SF1R135_RJ.FLR'
ELSEIF(NFLG.EQ.16) THEN
   NAMEOUT='RF1:SF1R0_RJ.FLR'
ELSEIF(NFLG.EQ.17) THEN
   NAMEOUT='RF1:SF1R45_RJ.FLR'
ELSEIF(NFLG.EQ.18) THEN
   NAMEOUT='RF1:SF1R90_RJ.FLR'
ELSEIF(NFLG.EQ.19) THEN
   NAMEOUT='RF1:SF1R135_RJ.FLR'
ELSEIF(NFLG.EQ.20) THEN
   NAMEOUT='RF1:SF1R0_RJ.FLR'
ELSEIF(NFLG.EQ.21) THEN
   NAMEOUT='RF1:SF1R45_RJ.FLR'
ELSEIF(NFLG.EQ.22) THEN
   NAMEOUT='RF1:SF1R90_RJ.FLR'
ELSEIF(NFLG.EQ.23) THEN
   NAMEOUT='RF1:SF1R135_RJ.FLR'
ELSEIF(NFLG.EQ.24) THEN
   NAMEOUT='RF1:SF1R0_RJ.FLR'
ELSEIF(NFLG.EQ.25) THEN
   NAMEOUT='RF1:SF1R45_RJ.FLR'
ELSEIF(NFLG.EQ.26) THEN
   NAMEOUT='RF1:SF1R90_RJ.FLR'
ELSEIF(NFLG.EQ.27) THEN
   NAMEOUT='RF1:SF1R135_RJ.FLR'
ELSEIF(NFLG.EQ.28) THEN
   NAMEOUT='RF1:SF1R0_RJ.FLR'
ELSEIF(NFLG.EQ.29) THEN
   NAMEOUT='RF1:SF1R45_RJ.FLR'
ELSEIF(NFLG.EQ.30) THEN
   NAMEOUT='RF1:SF1R90_RJ.FLR'
```

```
      ELSE
         NAMEOUT='RF1:SF1R135_RJ.FLR'
      ENDIF

      RETURN

      END




      SUBROUTINE SS_GABOR(GBR,N,M,NROT,NSC)

      INTEGER          R,C,RG,CG,RGPN,CGPN
      DIMENSION        NGBR(256,512),GBR(256,512)
      DIMENSION        TOP(512)
      COMPLEX          CMPLX,CNUM


C--   ESTABLISH CONTANTS AND PARAMETERS  --C

      PI=3.141592654

      FREQ=FLCAT(M)
      ANGLE=FLOAT(NROT)


C--   ESTABLISH THE CENTERING OF THE GABOR FUNCTION  --C

      RG=127-N/2
      RGPN=RG+N
      CG=255-N/2
      CGPN=CG+N


C--   ZERO OUT THE GBR TRANSFER DATA STRUCTURE  --C


      DO R=1,256
           DO C=1,512
              GBR(R,C)=0
           END DO
      END DO


C--   CONVERT FROM DEGREES TO RADIANS  --C

      THETA=ANGLE*PI/180.


      RG=127-N/2
      MNY=N/2-N


C--   CHOOSE BETWEEN COSINE OR SINE WAVE SELF SIMILAR GABOR FUNCITON  --C
```

```
         IF(NSC.EQ.1) THEN

C--   BEGIN GENERATING THE COSINE GABOR FUNCTION  --C

         DO  I=1,N

            CG=255-N/2
            MNX=N/2-N

            DO  J=1,N

C--   ESTABLISH THE ROTATION VARIABLE XP  --C

               XP=(FLOAT(MNX)*COS(THETA)-FLOAT(MNY)*SIN(THETA))

C--   ESTABLISH THE X AND Y VARIABLES  --C

               X=FLOAT(MNX)*FLOAT(MNX)
               Y=FLOAT(MNY)*FLOAT(MNY)

C--   GENERATE THE GABOR FUNCTION BASED ON THE XP, X, AND Y VARIABLES  --C

             GBR(RG,CG)=EXP(-(X+Y)/(2.*(FLOAT(N)/2./FREQ)*
     +       (FLOAT(N)/2./FREQ)))*
     +       (COS(2.*PI*XP*(FLOAT(N)/2./FREQ)))/
     +       (2.*PI*((FLOAT(N)/2./FREQ)**2))


            MNX=MNX+1
            CG=CG+1

            END DO

         MNY=MNY+1
         RG=RG+1

         END DO

C--   SCALE THE GABOR FUNCTION TO MAX EQUAL 255  --C

         DO C=1,512
            XTOP=0.
            DO R=1,256

                 XT2=GBR(R,C)
                 RTOP=XTOP-XT2
                  IF(RTOP.LT.0) THEN
                  XTOP=XT2
                  END IF
            END DO
            TOP(C)=XTOP

         END DO

         THE_MAX=0.
         DO C=1,512
            XT3=TOP(C)
            RTOP=THE_MAX-XT3
```

```
            IF(RTOP.LT.0) THEN
            THE_MAX=XT3
            END IF

         END DO

         DO R=1,256
            DO C=1,512
               GBR(R,C)=255.*(GBR(R,C)/THE_MAX)
            END DO
         END DO


C--   OR ELSE CREATE THE SINE WAVE SELF SIMILAR GABOR FUNCTION  --C


         ELSE

         DO I=1,N

            CG=255-N/2
            MNX=N/2-N

            DO J=1,N

               XP=(FLOAT(MNX)*COS(THETA)-FLOAT(MNY)*SIN(THETA))
               X=FLOAT(MNX)*FLOAT(MNX)
               Y=FLOAT(MNY)*FLOAT(MNY)
              GBR(RG,CG)=EXP(-(X+Y)/(2.*(FLOAT(N)/2./FREQ)*
     +        (FLOAT(N)/2./FREQ)))*
     +        (SIN(2.*PI*XP/(FLOAT(N)/2./FREQ)))/
     +        (2.*PI*((FLOAT(N)/2./FREQ)**2))
            MNX=MNX+1
            CG=CG+1

            END DO

         MNY=MNY+1
         RG=RG+1

         END DO

         DO C=1,512
            XTOP=0.
            DO R=1,256

                 XT2=GBR(R,C)
                 RTOP=XTOP-XT2
                  IF(RTOP.LT.0) THEN
                  XTOP=XT2
                  END IF
            END DO
            TOP(C)=XTOP

         END DO

         THE_MAX=0.
         DO C=1,512
            XT3=TOP(C)
            RTOP=THE_MAX-XT3
```

```
            IF(RTOP.LT.0) THEN
            THE_MAX=XT3
            END IF

         END DO

         DO R=1,256
            DO C=1,512
               GBR(R,C)=255.*(GBR(R,C)/THE_MAX)
            END DO
         END DO

         ENDIF

         RETURN
         END









         SUBROUTINE MSS_GABOR(GBR,N,M,M1,NROT,NSC)

         INTEGER          R,C,RG,CG,RGPN,CGPN
         DIMENSION        NGBR(256,512),GBR(256,512)
         DIMENSION        TOP(512)
         COMPLEX          CMPLX,CNUM


C--   ESTABLISH CONSTANTS AND PARAMETERS  --C

         PI=3.141592654

         FREQ=FLOAT(M)
         ANGLE=FLOAT(NROT)
         CYCLES=FLOAT(M1)

C--   ESTABLISH THE CENTERING OF THE GABOR FUNCTION  --C

         RG=127-N/2
         RGPN=RG+N
         CG=255-N/2
         CGPN=CG+N


C--   ZERO OUT THE GBR TRANSFER DATA STRUCTURE  --C


         DO R=1,256
               DO C=1,512
                  GBR(R,C)=0
               END DO
         END DO


C--   CONVERT FROM DEGREES TO RADIANS  --C
```

```
      THETA=ANGLE*PI/180.


      RG=127-N/2
      MNY=N/2-N


C--   CHOOSE BETWEEN COSINE OR SINE WAVE SELF SIMILAR GABOR FUNCITON  --C

      IF(NSC.EQ.1) THEN


C--   BEGIN GENERATING THE COSINE GABOR FUNCTION  --C

      DO  I=1,N

         CG=255-N/2
         MNX=N/2-N

         DO  J=1,N

C--   ESTABLISH THE ROTATION VARIABLE XP  --C

            XP=(FLOAT(MNX)*COS(THETA)-FLOAT(MNY)*SIN(THETA))

C--   ESTABLISH THE X AND Y VARIABLES  --C

            X=FLOAT(MNX)*FLOAT(MNX)
            Y=FLOAT(MNY)*FLOAT(MNY)

C--   GENERATE THE GABOR FUNCTION BASED ON THE XP, X, AND Y VARIABLES  --C

          GBR(RG,CG)=EXP(-(X+Y)/(2.*(FLOAT(N)/2./FREQ)*
     +       (FLOAT(N)/2./FREQ)))*
     +       (COS(2.*PI**CYCLES*XP*(FLOAT(N)/2./FREQ)))/
     +       (2.*CYCLES*PI*((FLOAT(N)/2./FREQ)**2))


         MNX=MNX+1
         CG=CG+1

         END DO

      MNY=MNY+1
      RG=RG+1

      END DO

C--   SCALE THE GABOR FUNCTION TO MAX EQUAL 255  --C

      DO C=1,512
         XTOP=0.
         DO R=1,256

              XT2=GBR(R,C)
              RTOP=XTOP-XT2
               IF(RTOP.LT.0) THEN
               XTOP=XT2
               END IF
```

```
            END DO
            TOP(C)=XTOP

         END DO

         THE_MAX=0.
         DO C=1,512
            XT3=TOP(C)
            RTOP=THE_MAX-XT3
            IF(RTOP.LT.0) THEN
            THE_MAX=XT3
            END IF

         END DO

         DO R=1,256
            DO C=1,512
               GBR(R,C)=255.*(GBR(R,C)/THE_MAX)
            END DO
         END DO


C--   OR ELSE CREATE THE SINE WAVE SELF SIMILAR GABOR FUNCTION  --C


         ELSE

         DO I=1,N

            CG=255-N/2
            MNX=N/2-N

            DO J=1,N

               XP=(FLOAT(MNX)*COS(THETA)-FLOAT(MNY)*SIN(THETA))
               X=FLOAT(MNX)*FLOAT(MNX)
               Y=FLOAT(MNY)*FLOAT(MNY)
             GBR(RG,CG)=EXP(-(X+Y)/(2.*(FLOAT(N)/2./FREQ)*
     +       (FLOAT(N)/2./FREQ)))*
     +       (SIN(2.*PI**CYCLES*XP/(FLOAT(N)/2./FREQ)))/
     +       (2.*PI*CYCLES*((FLOAT(N)/2./FREQ)**2))
            MNX=MNX+1
            CG=CG+1

            END DO

         MNY=MNY+1
         RG=RG+1

         END DO

         DO C=1,512
            XTOP=0.
            DO R=1,256

                  XT2=GBR(R,C)
                  RTOP=XTOP-XT2
                   IF(RTOP.LT.0) THEN
                   XTOP=XT2
                   END IF
```

```
            END DO
            TOP(C)=XTOP

         END DO

         THE_MAX=0.
         DO C=1,512
            XT3=TOP(C)
            RTOP=THE_MAX-XT3
            IF(RTOP.LT.0) THEN
            THE_MAX=XT3
            END IF

         END DO

         DO R=1,256
            DO C=1,512
               GBR(R,C)=255.*(GBR(R,C)/THE_MAX)
            END DO
         END DO

         ENDIF

         RETURN
         END









         SUBROUTINE NSS_GABOR(GBR,N,M,NROT,NSC)


         INTEGER          R,C,RG,CG,RGPN,CGPN
         DIMENSION        NGBR(256,512),GBR(256,512)
         DIMENSION        TOP(512)
         COMPLEX          CMPLX,CNUM

C--   ESTABLISH CONSTANTS AND PARAMETERS                    --C

         PI=3.141592654



         FREQ=FLOAT(M)
         ANGLE=FLOAT(NROT)

C--   ESTABLISH CENTERING OF GABOR FUNCTION                 --C

         RG=127-N/2
         RGPN=RG+N
         CG=255-N/2
         CGPN=CG+N


C-   ZERO OUT THE TRANSFER DATA STRUCTURE(S)                --C
```

```
         DO R=1,256
              DO C=1,512
                 GBR(R,C)=(0.,0.)
              END DO
         END DO

C--   CONVERT FROM DEGREES TO RADIANS                              --C

         THETA=ANGLE*PI/180.

         RG=127-N/2
         MNY=N/2-N


C--   CHOOSE BETWEEN SINE OR COSINE WAVE GABOR FUNCTION            --C

         IF(NSC.EQ.1) THEN


C--   BEGIN COSINE GABOR FUNCTION GENERATION                       --C

         DO I=1,N

            CG=255-N/2
            MNX=N/2-N

            DO J=1,N
          XP=(FLOAT(MNX)*COS(THETA)-FLOAT(MNY)*SIN(THETA))
          X=FLOAT(MNX)*FLOAT(MNX)
          Y=FLOAT(MNY)*FLOAT(MNY)
               GBR(RG,CG)=EXP(-(X+Y)/(2.*(FLOAT(N)/8.)*(FLOAT(N)/8.)))*
     +         COS(2.*PI*FREQ*XP)/(2.*PI*((FLOAT(N)/8.)**2))
            MNX=MNX+1
            CG=CG+1
            END DO
         MNY=MNY+1
         RG=RG+1
         END DO


C--   SCALE THE GABOR FUNCTION MAXIMUM TO EQUAL 255                --C
         DO C=1,512
            XTOP=0.
            DO R=1,256

                 XT2=GBR(R,C)
                 RTOP=XTOP-XT2
                  IF(RTOP.LT.0) THEN
                  XTOP=XT2
                  END IF
            END DO
            TOP(C)=XTOP

         END DO

         THE_MAX=0.
         DO C=1,512
            XT3=TOP(C)
            RTOP=THE_MAX-XT3
```

```
            IF(RTOP.LT.0) THEN
            THE_MAX=XT3
            END IF

         END DO

         DO R=1,256
            DO C=1,512
               GBR(R,C)=255.*(GBR(R,C)/THE_MAX)
            END DO
         END DO



C--   OR ELSE GENERATE THE SINE WAVE GABOR FUNCTION          --C


         ELSE

         DO I=1,N

            CG=255-N/2
            MNX=N/2-N

            DO J=1,N
          XP=(FLOAT(MNX)*COS(THETA)-FLOAT(MNY)*SIN(THETA))
          X=FLOAT(MNX)*FLOAT(MNX)
          Y=FLOAT(MNY)*FLOAT(MNY)
          GBR(RG,CG)=EXP(-(X+Y)/(2.*(FLOAT(N)/8.)*(FLOAT(N)/8.)))*
     +    SIN(2.*PI*FREQ*XP)/(2.*PI*((FLOAT(N)/8.)**2))
            MNX=MNX+1
            CG=CG+1
            END DO
         MNY=MNY+1
         RG=RG+1
         END DO

         DO C=1,512
            XTOP=0.
            DO R=1,256

                  XT2=GBR(R,C)
                  RTOP=XTOP-XT2
                   IF(RTOP.LT.0) THEN
                   XTOP=XT2
                   END IF
            END DO
            TOP(C)=XTOP

         END DO

         THE_MAX=0.
         DO C=1,512
            XT3=TOP(C)
            RTOP=THE_MAX-XT3
            IF(RTOP.LT.0) THEN
            THE_MAX=XT3
            END IF

         END DO
```

```
      DO R=1,256
         DO C=1,512
            GBR(R,C)=255.*(GBR(R,C)/THE_MAX)
         END DO
      END DO


      ENDIF

      RETURN
      END









      SUBROUTINE GABOR_TRANSFORM(GBR,NAR,NMAG,NR_NUM,NC_NUM,np_num)

      DIMENSION       NMAG(256,512),NAR(256,512)
      DIMENSION       GBR(256,512),TOP(512),XMAG(256,512)
      COMPLEX         U,CAR(256,512),CGBR(256,512)
      COMPLEX         CC(512),G(512),V
      INTEGER         R,C,RP,CP,RM1

      type*,'Begin gabor transform pass',np_num


C--   ZERO OUT THE TRANSFER VARIABLES AND DATA STRUCTURES  --C

      type*,'Zeroing out the transfer arrays'
      DO R=1,256
         DO C=1,512
            CC(C)=(0.,0.)
            G(C)=(0.,0.)
            CAR(R,C)=(0.,0.)
            CGBR(R,C)=(0.,0.)
            NMAG(R,C)=0
            TOP(C)=0.
         END DO
      END DO



      RP=9
      SUM=0.

      MFLG=1

      U=(1.0,0.0)
      V=(1.0,0.0)


C--   CONVERT DATA FROM INTEGER TO COMPLEX FORMAT  --C

      type*,'Converting the data to complex format'
```

```
      DO R=1,256

         DO C=1,512
            CAR(R,C)=NAR(R,C)*U
            CGBR(R,C)=GBR(R,C)*U

C--   COLUMN VECTORIZING OF DATA  --C

            CC(C)=CAR(R,C)
            G(C)=CGBR(R,C)
         END DO

C--   SEND DATA TO FAST FOURIER TRANSFORM SUBROUTINE  --C

            CALL FFT(CC,9,MFLG)
            CALL FFT(G,9,MFLG)

C--   PUT THE VECTORS BACK INTO MATRIX FORMAT  --C

         DO C=1,512
            CAR(R,C)=CC(C)
            CGBR(R,C)=G(C)
         END DO
      END DO


C--   ZERO OUT THE TRANSFER STRUCTURES  --C

      DO C=1,512
         CC(C)=(0.,0.)
         G(C)=(0.,0.)
      END DO

C--   ROW VECTORIZE THE COLUMN FFT'd DATA  --C

      DO C=1,512
         DO R=1,256
            CC(R)=CAR(R,C)
            G(R)=CGBR(R,C)
         END DO

C--   SEND DATA TO FFT SUBROUTINE  --C

            CALL FFT(CC,8,MFLG)
            CALL FFT(G,8,MFLG)
         DO R=1,256

C-- CONVERT VECTORS BACK TO MATRIX FORMAT  --C

            CAR(R,C)=CC(R)
            CGBR(R,C)=G(R)
         END DO

      END DO


C--   COMPLETED THE TWO DIMENSIONAL FFT OF IMAGE AND GABOR FUNCTION  --C

C--   PERFORM POINT FOR POINT MULTIPLICATION OF THE  --C
C--   FFT IMAGE WITH THE COMPLEX CONJUGATE OF THE    --C
```

```
C--   FFT GABOR FUNCTION.                                    --C

         DO R=1,256
            DO C=1,512
               CAR(R,C)=CAR(R,C)*CONJG(CGBR(R,C))
            END DO
         END DO


C--   BEGIN INVERSE FFT OF CORRELATION  --C

         DO R=1,256
            DO C=1,512

C--   COLUMN VECTORIZE THE CORRELATION IMAGE  --C

               CC(C)=CAR(R,C)

            END DO

C--   SEND VECTOR(S) TO INVERSE FFT ROUTINE  --C

            CALL FFT(CC,9,MFLG)

C--   PUT VECTORS BACK INTO MATRIX FORMAT  --C

            DO C=1,512
               CAR(R,C)=CC(C)
            END DO

         END DO

C--   ZERO OUT THE TRANSFER DATA STRUCTURE  --C

         DO C=1,512
            CC(C)=(0.,0.)
         END DO

C--   ROW VECTORIZE THE COLUMN FFT'd DATA  --C

         DO C=1,512
            DO R=1,256
               CC(R)=CAR(R,C)
            END DO

C--   SEND VECTOR(S) TO INVERSE FFT ROUTINE  --C

            CALL FFT(CC,8,MFLG)

C--   PUT VECTOR(S) BACK INTO MATRIX FORMAT  --C

            DO R=1,256
               CAR(R,C)=CC(R)
            END DO

         END DO

C--   INVERSE FFT OPERATION IS A FORWARD FFT OPERATION WITH  --C
C--   THE CONJUGATE OF THE RESULTANT IMAGE MULTIPLIED BY THE --C
C--   DIMENSIONS OF THE IMAGE (X BY Y).                      --C
```

```
      DO R=1,256
         DO C=1,512
         CAR(R,C)=256*512*CONJG(CAR(R,C))

C--   CONVERT THE IMAGE INTO A MAGNITUDE SQUARED, OR INTENSITY  --C

         XMAG(R,C)=CABS(CAR(R,C))*CABS(CAR(R,C))

         END DO

      END DO


C--   SCALE THE ENTIRE IMAGE TO MAX EQUAL 255  --C

      DO C=1,512
         XTOP=0.
         DO R=1,256

              XT2=XMAG(R,C)
              RTOP=XTOP-XT2
               IF(RTOP.LT.0) THEN
               XTOP=XT2
               END IF
         END DO
         TOP(C)=XTOP

      END DO

      THE_MAX=0.
      DO C=1,512
         XT3=TOP(C)
         RTOP=THE_MAX-XT3
         IF(RTOP.LT.0) THEN
         THE_MAX=XT3
         END IF
      END DO
      DO R=1,256
         DO C=1,512
            XMAG(R,C)=255.*(XMAG(R,C)/THE_MAX)
            NMAG(R,C)=IFIX(XMAG(R,C))
         END DO
      END DO


C--   BEGIN ROTATION OF IMAGE ABOUT THE X AND Y AXES  --C

      DO R=1,128
         RP=R+128
         DO C=1,256
            CP=C+256
            NTEMP1=NMAG(R,C)
            NMAG(R,C)=NMAG(RP,CP)
            NMAG(RP,CP)=NTEMP1
            NTEMP2=NMAG(R,CP)
            NMAG(R,CP)=NMAG(RP,C)
            NMAG(RP,C)=NTEMP2
         END DO
      END DO
```

```
      DO R=1,128
         NR=257-R
         DO C=1,512
            NTEMP1=NMAG(R,C)
            NMAG(R,C)=NMAG(NR,C)
            NMAG(NR,C)=NTEMP1
         END DO
      END DO
      DO C=1,256
         NC=513-C
         DO R=1,256
            NTEMP1=NMAG(R,C)
            NMAG(R,C)=NMAG(R,NC)
            NMAG(R,NC)=NTEMP1
         END DO
      END DO

C--   ROTATION COMPLETED  --C

      type*,'completed gabor transform pass',np_num
      RETURN
      END









      SUBROUTINE PUT_FLIR_IM(NAMEOUT,NMAG,ROW_NUM,COL_NUM)

      IMPLICIT          INTEGER(A-Z)
      DIMENSION         NMAG(256,512)
      BYTE              NMAG_BYTE(240,640)
      CHARACTER*24      NAMEOUT

C--   ZERO OUT THE BYTE FORMAT TRANSFER DATA STRUCTURE  --C

      DO R=1,240
         DO C=1,640
            NMAG_BYTE(R,C)=0
         END DO
      END DO

C--   PUT THE 256 BY 512 IMAGE INTO A 240 BY 640 FORMAT  --C


      RP=9
      DO R=1,240
         DO C=65,576
            CP=C-64
              NMAG_BYTE(R,C)=MOD(NMAG(RP,CP),256)-128
         END DO
      RP=RP+1
      END DO

      OPEN(20,FILE=NAMEOUT,STATUS='NEW',FORM='UNFORMATTED',
```

```
     +   RECL=COL_NUM/4+1,RECORDTYPE='FIXED')

         DO 100 R=1,ROW_NUM
            WRITE(20) (NMAG_BYTE(R,C),C=1,COL_NUM)
100      CONTINUE

C--   ZERO OUT THE DATA STRUCTURES  --C


         CLOSE(20)
         RETURN
         END









         SUBROUTINE FFT(F,LN,MFLG)

         COMPLEX          F(1024),U,W,T,CMPLX
         PI=3.141592654
         N=2**LN
         NV2=N/2
         NM1=N-1
         J=1
         DO 3 I=1,NM1
            IF(I.GE.J) GO TO 1
               T=F(J)
               F(J)=F(I)
               F(I)=T
1        K=NV2
2           IF(K.GE.J) GO TO 3
               J=J-K
               K=K/2
               GO TO 2
3              J=J+K
         DO 5 L=1,LN
            LE=2**L
            LE1=LE/2
            U=(1.0,0.0)
            W=CMPLX(COS(PI/LE1),-SIN(PI/LE1))
            DO 5 J=1,LE1
               DO 4 I=J,N,LE
                  IP=I+LE1
                  T=F(IP)*U
                  F(IP)=F(I)-T
4                 F(I)=F(I)+T
5              U=U*W
         DO 6 I=1,N
6           F(I)=F(I)/FLOAT(N)
         RETURN
         END
```

```
      subroutine help



1     continue

      DO I=1,45
      TYPE*,' '
      END DO
      type*,'WELCOME TO THE GABOR TRANSFORM HELP SCREEN'
      TYPE*,' '
      TYPE*,' '
      TYPE*,'************************************************'
      TYPE*,'*                                              *'
      TYPE*,'*    NAME OF FLIR IMAGE      -   1             *'
      TYPE*,'*                                              *'
      TYPE*,'*    LOWEST FREQUENCY        -   2             *'
      TYPE*,'*                                              *'
      TYPE*,'*    HIGHEST FREQUENCY       -   3             *'
      TYPE*,'*                                              *'
      TYPE*,'*    STANDARD DEVIATION      -   4             *'
      TYPE*,'*    AT MAX EXTENT                             *'
      TYPE*,'*                                              *'
      TYPE*,'*    MAX ANGULAR ROTATION    -   5             *'
      TYPE*,'*                                              *'
      TYPE*,'*    INCR. OF ROTATION       -   6             *'
      TYPE*,'*                                              *'
      TYPE*,'*    SIZE OF GABOR WINDOW    -   7             *'
      TYPE*,'*                                              *'
      TYPE*,'*    REGULAR SELF SIMILAR    -   8             *'
      TYPE*,'*    MODIFIED SELF SIMILAR   -   9             *'
      TYPE*,'*    NON-SELF SIMILAR        -   10            *'
      TYPE*,'*                                              *'
      TYPE*,'*    SEGMENTATION            -   11            *'
      TYPE*,'*    CLASSIFICATION          -   12            *'
      TYPE*,'*                                              *'
      TYPE*,'*    SINE                    -   13            *'
      TYPE*,'*    COSINE                  -   14            *'
      TYPE*,'*                                              *'
      TYPE*,'*    RETURN TO MAIN PROGRAM  -   15            *'
      TYPE*,'************************************************'
      TYPE*,' '
      TYPE*,'ENTER YOUR CHOICE: '
      READ(5,*)NHELP
      IF(NHELP.EQ.1) THEN
      TYPE*,'The name of the FLIR image you wish to transform'
      type*,'is a user specified name.  The name can be any'
      type*,'extension provided the file the name refers to is'
      type*,'of the proper format (240 by 640, BYTE).'
      type*,' '
      type*,'To return to the menu, enter a 1'
      read(5,*)n
      goto 1
      ELSEIF(NHELP.EQ.2) THEN
```

```
type*,'The lowest frequency is the lower bound on frequencies '
type*,'the user wishes to investigate.  the difference between '
type*,'the highest and lowest frequencies of interest plus one '
type*,'establishes the number of frequency passes through the '
type*,'program.  This value, coupled with the number of  '
type*,'orientations, will determine the total number of passes '
type*,'through the program. '
type*,' '
type*,' '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
goto 1
ELSEIF(NHELP.EQ.3) THEN
type*,'The highest frequency is the highest number of cycles '
type*,'per envelope the user wishes to investigate.  The  '
type*,'difference between the highest and loest frequencies '
type*,'plus one determines the number of frequency passes '
type*,'through the program.  This value, coupled with the  '
type*,'number of orientations determines the total number of '
type*,'passes through the program. '
type*,' '
type*,' '
type*,' '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
goto 1

ELSEIF(NHELP.EQ.4) THEN
type*,'The standard deviation at max extent is the number '
type*,'of standard deviations the user wishes to correspond '
type*,'to the window size parameter. '
type*,'For example:  If 16 is chosen for window size, and '
type*,'four is chosen for std. dev., then four standard  '
type*,'deviations will correspond to the N/2 pixel from the '
type*,'center of the gabor window.  For this example, the eighth '
type*,'pixel from the center of the Gabor function will correspond '
type*,'to four standard deviations. '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
goto 1

ELSEIF(NHELP.EQ.5) THEN
type*,'Maximum angular orientation corresponds to the maximum '
type*,'angular rotation (in degrees) the user wishes to  '
type*,'investigate.  The maximum angular orientation MUST be '
type*,'an integer multiple of the increments of rotation. '
type*,' '
type*,'For example:  If the user wishes to investigate the '
type*,'processing characteristics of Gabor functions from 0 to '
type*,'135 degrees.  The increment of rotation MUST divide into '
type*,'135 evenly. '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
goto 1

ELSEIF(NHELP.EQ.6) THEN
```

```
type*,'Increment of rotation is the change in orientation  '
type*,'relative to 0 degrees.  The increment of change MUST '
type*,'evenly divide into the maximum angular orientation. '
type*,' '
type*,'For example:  if Max. Orient. equals 135, then the '
type*,'Incr. of Rot. MUST be 5, 10, 15, so on up to, and '
type*,'including 135. '
type*,' '
type*,' '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
goto 1

ELSEIF(NHELP.EQ.7) THEN
type*,'Size of the Gabor window is a parameter linked to '
type*,'the standard deviation.  Given a std. dev., the Gabor '
type*,'window size (N) divided by 2 (N/2) will correspond '
type*,'to the std dev. '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
goto 1

ELSEIF(NHELP.EQ.8) THEN
type*,'The regular self similar Gabor function produces '
type*,'a single cycle per envelope Gabor function with '
type*,'the specific orientation determined by the user. '
type*,'As the function's spread becomes smaller, the '
type*,'frequency increases to compensate, thereby maintaining'
type*,'the number of cycles constant under the Gaussian'
type*,'envelope.'
type*,'The regular self similar Gabor function has '
type*,'biological implications since, it is believed '
type*,'that the brain possesses cells which respond in the '
type*,'same fashion as Gabor functions. '
type*,' '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
goto 1

ELSEIF(NHELP.EQ.9) THEN
type*,'Modified Gabor functions possess the dilating nature '
type*,'of the regular self similar Gabor function but allows '
type*,'the user to specify the number of cycles per envelope. '
type*,' '
type*,'This allows the modified Gabor function to take on the '
type*,'characteristic of non-self similar Gabor functions. '
type*,'The non-self similar Gabor function information may '
type*,'be found under number 10. '
type*,' '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
```

```
goto 1

ELSEIF(NHELP.EQ.10) THEN
type*,'The non-self similar Gabor function allows the user to  '
type*,'specify the number of cycles per envelope but fixes the '
type*,'standard deviation of the Gaussian cofactor. '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
goto 1

ELSEIF(NHELP.EQ.11) THEN
type*,'Segmentation uses, primarily, the non-self similar nature '
type*,'of the Gabor functions (be it explicit by using non-self '
type*,'similar Gabor functions or implicit when using the modified '
type*,'Gabor function.  Segmentation is the act of separating an '
type*,'object of interest from regions of relatively no interest. '
type*,'The regions of relatively no interest may have significant '
type*,'structure thereby cluttering the image(s). '
type*,' '
type*,' '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
goto 1

ELSEIF(NHELP.EQ.12) THEN
type*,'Classification is the act of determining the target '
type*,'or object of interest identification (signature, response,'
type*,'or shape, color, texture, etc.).  This program generates '
type*,'the jet vector components needed for classification. '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
goto 1

ELSEIF(NHELP.EQ.13) THEN
type*,'Sine wave Gabor functions primarily detect edges. '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
```

```
goto 1

ELSEIF(NHELP.EQ.14) THEN
type*,'Cosine wave Gabor functions primarily fills in bodies. '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,' '
type*,'To return to the menu, enter a 1'
read(5,*)n
goto 1

ELSE

endif
return
end
```

Appendix D. Multiple image superposition FORTRAN source code.

THRESH_ADD_HIST.FOR

SYNOPSIS: The THRESH_ADD_HIST.FOR program takes user defined images and adds them together, pixel for pixel. The program generates an output file for the superimposed image and prompts the user for labeling or not labeling highest pixel value locations. After superimposing the images, the program generates a histogram of the normalized intensity distribution of the superimposed image. A threshold is set based on the normalized intensity distribution. After thresholding the superimposed image, the pixel values above the threshold level are set to unity and those below the threshold are set to zero. The program continues until the user desires termination.

INPUT: Images, 240 by 640 pixels.

OUTPUTS: Superimposed Gabor image(s)
Thresholded, binarized versions of superimposed Gabor image(s).
Histogram of the normalized intensity distribution(s).
OPTIONAL: Locations of all highest pixel values.

```
C-----------------------------------------------------------------------C
C
C       PROGRAM TO SUPERIMPOSE IMAGES, HISTOGRAM THE SUPERIMPOSED IMAGE
C       BASED ON INTENSITY, THEN THRESHOLD THE IMAGE BASED ON THE
C       INTENSITY DISTRIBUTION.
C
C       ARRAYS USED:
C
C               NAR             - INTEGER FORMAT OF THE INPUT IMAGE
C               NSUM            - INTEGER FORMAT OF THE SUPERIMPOSED
C                                 IMAGES
C               COUNT           - NOT USED
C               RNAR            - REAL FORMAT OF INPUT IMAGE
C               RSUM            - REAL FORMAT OF THE SUPERIMPOSED IMAGES
C               TOP             - MAXIMUM PIXEL VALUES PER COLUMN
C
C       VARIABLES USED:
C
C               NUM             - NUMBER OF FLIR IMAGES TO ADD
C               NLABEL          - LABEL TOGGLE, IF NLABEL=1, THEN LABEL
C                                 ALL PIXELS PASSING A CERTAIN THRESHOLD
C                                 (LOCATIONS)
C               XMAX,XT2,XDIFF, - USED FOR THE PURPOSES OF FINDING THE
C               XDIFF2, XTOP      MAXIMUM PIXEL VALUE FOR SCALING
C               NLO             - LOWER THRESHOLD LEVEL
C               NHI             - UPPER THRESHOLD LEVEL
C               NCOUNT          - TOTAL NUMBER OF LABELED PIXELS
C               NCHOICE         - REPEAT THE PROCESS INDICATOR
C
C       ASCII VARIABLES USED:
C
C               NAME            - NAME OF INPUT FILE(S)
C               NAMEOUT         - NAME OF THE NON-THRESHOLDED OUTPUT FILE
C               THR_NAME        - NAME OF THE THRESHOLDED OUTPUT FILE
C
C       WRITTEN BY:  KEVIN W. AYER, CAPT, USAF          DATE:  15 JUL 89
C
C       UPDATED BY:  KEVIN W. AYER, CAPT, USAF          DATE:  31 JUL 89
C
C       UPDATED BY:                                     DATE:
C
C
C-----------------------------------------------------------------------C



      PROGRAM MAIN_ADD
      DIMENSION       NAR(240,640),NSUM(240,640),COUNT(53)
      DIMENSION       RNAR(240,640),RSUM(240,640),TOP(640)
      CHARACTER*24    NAME,NAMEOUT,THR_NAME


1     CONTINUE
```

```
      TYPE*,'HOW MANY FLIR IMAGES DO YOU WISH TO SUM?'
      READ(5,*)NUM
      TYPE*,'WHAT DO YOU WISH TO CALL THE SUPERIMPOSED IMAGE FILE?'
      ACCEPT 40,NAMEOUT
      TYPE*,'DO YOU WISH TO LABEL TO HIGHEST PIXELS? NO=0, YES=1'
      READ(5,*)NLABEL
      TYPE*,'ZEROING OUT THE TRANSFER ARRAYS'
      DO I=1,240
         DO J=1,640
              NSUM(I,J)=0
              RSUM(I,J)=0.
         END DO
      END DO
      TYPE*,'BEGINNING THE SUMMING ROUTINE'
      DO 45 L=1,NUM

      TYPE*,'NAME OF FLIR IMAGE ',L,'    ?'
      ACCEPT 40,NAME

      CALL GET_FLIR_IM(NAME,NAR,240,640)

      DO I=1,240
         DO J=1,640

            NSUM(I,J)=NSUM(I,J)+NAR(I,J)
            RSUM(I,J)=FLOAT(NSUM(I,J))

         END DO
      END DO


45    CONTINUE
      TYPE*,'SUMMING ROUTINE COMPLETED'
      TYPE*,' '
      TYPE*,'FINDING THE MAX VALUE AND SCALING TO THIS VALUE'
      DO I=1,240
      XMAX=0.
         DO J=17,528
            XT2=RSUM(I,J)
            XDIFF=XMAX-XT2
            IF(XDIFF.LT.0) THEN
              XMAX=XT2
            ENDIF
         END DO
         TOP(I)=XMAX
      END DO
      XTOP=0.
      DO I=1,240
         XT3=TOP(I)
         XDIFF2=XTOP-XT3
         IF(XDIFF2.LT.0) THEN
            XTOP=XT3

         END IF
      END DO
      DO I=1,240
         DO J=1,640
            RSUM(I,J)=RSUM(I,J)*255./XTOP
            NSUM(I,J)=IFIX(RSUM(I,J))
```

```
         END DO
       END DO
       OPEN(31,FILE='LINES.DAT',STATUS='NEW',FORM='FORMATTED')
       WRITE(31,131)
131    FORMAT(1X,'COL 200',5X,'COL 400')
       DO I=1,240
       WRITE(31,*)NSUM(I,200),NSUM(I,400)
       END DO
       WRITE(31,132)
       DO J=1,640
       WRITE(31,*)NSUM(92,J),NSUM(176,J)
       END DO

132    FORMAT(1X,'ROW 92',5X,'ROW 176')
       CLOSE(31)
       TYPE*,'SCALING COMPLETED'
       TYPE*,' '
       TYPE*,' '
       TYPE*,'BEGIN HISTOGRAM'
        CALL I_HIST(NSUM,NLO,NHI,NLOB,NHIB)
       TYPE*,'HISTOGRAM COMPLETED'

       CALL PUT_FLIR_IM(NAMEOUT,NSUM,240,640)

       TYPE*,'THRESHOLDING IMAGE BASED ON HISTOGRAM'

             NLO=NLO-1.
             NHI=NHI+1.
             IF(NLO.LT.0) THEN
                NLO=0
             END IF
             IF(NHI.GT.255) THEN
                NHI=255
             END IF


             TYPE*,'LOWER THRESHOLD = ',NLO,'BIN #',NLOB
             TYPE*,'UPPER THRESHOLD = ',NHI,'BIN #',NHIB

               DO I=1,240
                  DO J=1,640
               IF(NSUM(I,J).GT.255) THEN
                  NSUM(I,J)=255
               END IF
                  IF(NSUM(I,J).LT.NHI.AND.NSUM(I,J).GT.NLO) THEN

                        NSUM(I,J)=128

                  ELSE
                        NSUM(I,J)=127
                  END IF
                  END DO

               END DO

       TYPE*,'THRESHOLDING COMPLETED'
       TYPE*,' '
       TYPE*,' '
```

```
              TYPE*,'WHAT DO YOU WISH TO CALL THE THRESHOLDED '
              TYPE*,'VERSION OF THE SUPERIMPOSED IMAGE?'
              ACCEPT 40,THR_NAME




               IF(NLABEL.EQ.1) THEN
          TYPE*,'LABELING THE HIGH PIXELS FOR THEIR LOCATION'
          OPEN(42,FILE='LAB_FLR_PIX.DAT',STATUS='NEW',
     +    FORM='FORMATTED')
                  WRITE(42,141)THR_NAME
                  NCOUNT=1
                  DO I=1,240
                     DO J=65,576
                        IF(NSUM(I,J).EQ.127) THEN
                  WRITE(42,142)I,J
                        END IF
                     NCOUNT=NCOUNT+1
                     END DO
                  NCOUNT=NCOUNT+1
                  END DO
                  WRITE(42,143)NCOUNT
          END IF
141       FORMAT(1X,'FILENAME:  ',A24,//,1X,'  ROW',
     +    5X,'COLUMN')
142       FORMAT(1X,I5,5X,I5)
143       FORMAT(1X,//,1X,'TOTAL NUMBER OF PIXELS: ',I10)
          CLOSE(42)
          TYPE*,'LABELING COMPLETED'

          TYPE*,'STORING THRESHOLDED VERSION OF ',NAMEOUT

                  CALL PUT_FLIR_IM(THR_NAME,NSUM,240,640)



          TYPE*,'DO YOU WISH TO ADD OTHER IMAGES TOGETHER?'
          TYPE*,' NO=0, YES=1'
          READ(5,*)NCHOICE
          IF(NCHOICE.EQ.1) THEN
            GOTO 1
          ELSE
          ENDIF


40        FORMAT(A24)
50        FORMAT(A3)
          END
```

```
C-----------------------------------------------------------------------C
C
C       SUBROUTINE TO HISTOGRAM THE SUPERIMPOSED IMAGE
C
C       ARRAYS USED:
C
C               NAR             - INTEGER FORMAT OF INPUT FILE
C               HIST            - HISTOGRAM VALUES
C               RLHIST          - LOGARITHM OF THE HISTOGRAM VALUES
C
C       VARIABLES USED:
C
C               NCT*            - BIN TOTAL COUNTERS
C               TOP,DIFF1,DIFF2 - VARIABLES ASSOCIATED WITH FINDING THE
C               DIFF3,DIFF4       HIGH AND LOW THRESHOLD BOUNDS
C               DIFF5
C               NLO,NLOB        - LOWER THRESHOLD LEVEL
C               NHI,NHIB        - UPPER THRESHOLD LEVEL
C               NLOJ            - LOWER THRESHOLD LEVEL WHEN THE MAX
C                                 BIN COUNT IS NOT THE LOWER THRESHOLD LEVEL
C
C       ASCII VARIABLES USED:
C
C               NAMEIN          - NAME OF THE INPUT FILE
C               MATRIX          - NAME OF THE NON-LOGARITHM HISTOGRAM
C                                 FILE FOR MATRIX-X OUTPUT
C               LOGMAT          - NAME OF THE LOGARITHM HISTOGRAM FILE
C                                 FOR MATRIX-X OUTPUT
C
C       WRITTEN BY:  KEVIN W. AYER, CAPT, USAF          DATE:  15 JUL 89
C
C       UPDATED BY:                                     DATE:
C
C
C-----------------------------------------------------------------------C


       SUBROUTINE I_HIST(NAR,NLO,NHI,NLOB,NHIB)
       DIMENSION NAR(240,640)
       DIMENSION RLHIST(53,1),HIST(53,1)
       INTEGER R,C

       CHARACTER*24 NAMEIN,MATRIX,LOGMAT

 1     CONTINUE
       DUMMY=0.

       TYPE*,' '
       TYPE*,'THRESHOLD FOR COUNT LEVEL?'
       READ(5,*)TOP1
       TYPE*,'FOR MATRIX-X LINEAR OUTPUT, GIVE FILENAME?'
       ACCEPT 40,MATRIX
       TYPE*,'FOR MATRIX-X LOGARITHMIC OUTPUT, GIVE FILENAME?'
       ACCEPT 40,LOGMAT
 40    FORMAT(A24)

       DO I=1,53
          HIST(I,1)=0.
       END DO
```

```
NCT0=0
NCT1=0
NCT2=0
NCT3=0
NCT4=0
NCT5=0
NCT6=0
NCT7=0
NCT8=0
NCT9=0
NCT10=0
NCT11=0
NCT12=0
NCT13=0
NCT14=0
NCT15=0
NCT16=0
NCT17=0
NCT18=0
NCT19=0
NCT20=0
NCT21=0
NCT22=0
NCT23=0
NCT24=0
NCT25=0
NCT26=0
NCT27=0
NCT28=0
NCT29=0
NCT30=0
NCT31=0
NCT32=0
NCT33=0
NCT34=0
NCT35=0
NCT36=0
NCT37=0
NCT38=0
NCT39=0
NCT40=0
NCT41=0
NCT42=0
NCT43=0
NCT44=0
NCT45=0
NCT46=0
NCT47=0
NCT48=0
NCT49=0
NCT50=0
NCT51=0
NCT52=0

DO R=1,240
   DO C=1,640
        IF(NAR(R,C).EQ.0) THEN
                NCT0=NCT0+1
```

```
ELSEIF(NAR(R,C).GE.1.AND.NAR(R,C).LT.5) THEN
        NCT1=NCT1+1
ELSEIF(NAR(R,C).GE.5.AND.NAR(R,C).LT.10) THEN
        NCT2=NCT2+1
ELSEIF(NAR(R,C).GE.10.AND.NAR(R,C).LT.15) THEN
        NCT3=NCT3+1
ELSEIF(NAR(R,C).GE.15.AND.NAR(R,C).LT.20) THEN
        NCT4=NCT4+1
ELSEIF(NAR(R,C).GE.20.AND.NAR(R,C).LT.25) THEN
        NCT5=NCT5+1
ELSEIF(NAR(R,C).GE.25.AND.NAR(R,C).LT.30) THEN
        NCT6=NCT6+1
ELSEIF(NAR(R,C).GE.30.AND.NAR(R,C).LT.35) THEN
        NCT7=NCT7+1
ELSEIF(NAR(R,C).GE.35.AND.NAR(R,C).LT.40) THEN
        NCT8=NCT8+1
ELSEIF(NAR(R,C).GE.40.AND.NAR(R,C).LT.45) THEN
        NCT9=NCT9+1
ELSEIF(NAR(R,C).GE.45.AND.NAR(R,C).LT.50) THEN
        NCT10=NCT10+1
ELSEIF(NAR(R,C).GE.50.AND.NAR(R,C).LT.55) THEN
        NCT11=NCT11+1
ELSEIF(NAR(R,C).GE.55.AND.NAR(R,C).LT.60) THEN
        NCT12=NCT12+1
ELSEIF(NAR(R,C).GE.60.AND.NAR(R,C).LT.65) THEN
        NCT13=NCT13+1
ELSEIF(NAR(R,C).GE.65.AND.NAR(R,C).LT.70) THEN
        NCT14=NCT14+1
ELSEIF(NAR(R,C).GE.70.AND.NAR(R,C).LT.75) THEN
        NCT15=NCT15+1
ELSEIF(NAR(R,C).GE.75.AND.NAR(R,C).LT.80) THEN
        NCT16=NCT16+1
ELSEIF(NAR(R,C).GE.80.AND.NAR(R,C).LT.85) THEN
        NCT17=NCT17+1
ELSEIF(NAR(R,C).GE.85.AND.NAR(R,C).LT.90) THEN
        NCT18=NCT18+1
ELSEIF(NAR(R,C).GE.90.AND.NAR(R,C).LT.95) THEN
        NCT19=NCT19+1
ELSEIF(NAR(R,C).GE.95.AND.NAR(R,C).LT.100) THEN
        NCT20=NCT20+1
ELSEIF(NAR(R,C).GE.100.AND.NAR(R,C).LT.105) THEN
        NCT21=NCT21+1
ELSEIF(NAR(R,C).GE.105.AND.NAR(R,C).LT.110) THEN
        NCT22=NCT22+1
ELSEIF(NAR(R,C).GE.110.AND.NAR(R,C).LT.115) THEN
        NCT23=NCT23+1
ELSEIF(NAR(R,C).GE.115.AND.NAR(R,C).LT.120) THEN
        NCT24=NCT24+1
ELSEIF(NAR(R,C).GE.120.AND.NAR(R,C).LT.125) THEN
        NCT25=NCT25+1
ELSEIF(NAR(R,C).GE.125.AND.NAR(R,C).LT.130) THEN
        NCT26=NCT25+1
ELSEIF(NAR(R,C).GE.130.AND.NAR(R,C).LT.135) THEN
        NCT27=NCT27+1
ELSEIF(NAR(R,C).GE.135.AND.NAR(R,C).LT.140) THEN
        NCT28=NCT28+1
ELSEIF(NAR(R,C).GE.140.AND.NAR(R,C).LT.145) THEN
        NCT29=NCT29+1
ELSEIF(NAR(R,C).GE.145.AND.NAR(R,C).LT.150) THEN
        NCT30=NCT30+1
```

```
            ELSEIF(NAR(R,C).GE.150.AND.NAR(R,C).LT.155) THEN
                    NCT31=NCT31+1
            ELSEIF(NAR(R,C).GE.155.AND.NAR(R,C).LT.160) THEN
                    NCT32=NCT32+1
            ELSEIF(NAR(R,C).GE.160.AND.NAR(R,C).LT.165) THEN
                    NCT33=NCT33+1
            ELSEIF(NAR(R,C).GE.165.AND.NAR(R,C).LT.170) THEN
                    NCT34=NCT34+1
            ELSEIF(NAR(R,C).GE.170.AND.NAR(R,C).LT.175) THEN
                    NCT35=NCT35+1
            ELSEIF(NAR(R,C).GE.175.AND.NAR(R,C).LT.180) THEN
                    NCT36=NCT36+1
            ELSEIF(NAR(R,C).GE.180.AND.NAR(R,C).LT.185) THEN
                    NCT37=NCT37+1
            ELSEIF(NAR(R,C).GE.185.AND.NAR(R,C).LT.190) THEN
                    NCT38=NCT38+1
            ELSEIF(NAR(R,C).GE.190.AND.NAR(R,C).LT.195) THEN
                    NCT39=NCT39+1
            ELSEIF(NAR(R,C).GE.195.AND.NAR(R,C).LT.200) THEN
                    NCT40=NCT40+1
            ELSEIF(NAR(R,C).GE.200.AND.NAR(R,C).LT.205) THEN
                    NCT41=NCT41+1
            ELSEIF(NAR(R,C).GE.205.AND.NAR(R,C).LT.210) THEN
                    NCT42=NCT42+1
            ELSEIF(NAR(R,C).GE.210.AND.NAR(R,C).LT.215) THEN
                    NCT43=NCT43+1
            ELSEIF(NAR(R,C).GE.215.AND.NAR(R,C).LT.220) THEN
                    NCT44=NCT44+1
            ELSEIF(NAR(R,C).GE.220.AND.NAR(R,C).LT.225) THEN
                    NCT45=NCT45+1
            ELSEIF(NAR(R,C).GE.225.AND.NAR(R,C).LT.230) THEN
                    NCT46=NCT46+1
            ELSEIF(NAR(R,C).GE.230.AND.NAR(R,C).LT.235) THEN
                    NCT47=NCT47+1
            ELSEIF(NAR(R,C).GE.235.AND.NAR(R,C).LT.240) THEN
                    NCT48=NCT48+1
            ELSEIF(NAR(R,C).GE.240.AND.NAR(R,C).LT.245) THEN
                    NCT49=NCT49+1
            ELSEIF(NAR(R,C).GE.245.AND.NAR(R,C).LT.250) THEN
                    NCT50=NCT50+1
            ELSEIF(NAR(R,C).GE.250.AND.NAR(R,C).LT.255) THEN
                    NCT51=NCT51+1
            ELSE
                    NCT52=NCT52+1
            ENDIF
    END DO
END DO

HIST(1,1)=FLOAT(NCT0)
HIST(2,1)=FLOAT(NCT1)
HIST(3,1)=FLOAT(NCT2)
HIST(4,1)=FLOAT(NCT3)
HIST(5,1)=FLOAT(NCT4)
HIST(6,1)=FLOAT(NCT5)
HIST(7,1)=FLOAT(NCT6)
HIST(8,1)=FLOAT(NCT7)
HIST(9,1)=FLOAT(NCT8)
HIST(10,1)=FLOAT(NCT9)
HIST(11,1)=FLOAT(NCT10)
HIST(12,1)=FLOAT(NCT11)
```

```
HIST(13,1)=FLOAT(NCT12)
HIST(14,1)=FLOAT(NCT13)
HIST(15,1)=FLOAT(NCT14)
HIST(16,1)=FLOAT(NCT15)
HIST(17,1)=FLOAT(NCT16)
HIST(18,1)=FLOAT(NCT17)
HIST(19,1)=FLOAT(NCT18)
HIST(20,1)=FLOAT(NCT19)
HIST(21,1)=FLOAT(NCT20)
HIST(22,1)=FLOAT(NCT21)
HIST(23,1)=FLOAT(NCT22)
HIST(24,1)=FLOAT(NCT23)
HIST(25,1)=FLOAT(NCT24)
HIST(26,1)=FLOAT(NCT25)
HIST(27,1)=FLOAT(NCT26)
HIST(28,1)=FLOAT(NCT27)
HIST(29,1)=FLOAT(NCT28)
HIST(30,1)=FLOAT(NCT29)
HIST(31,1)=FLOAT(NCT30)
HIST(32,1)=FLOAT(NCT31)
HIST(33,1)=FLOAT(NCT32)
HIST(34,1)=FLOAT(NCT33)
HIST(35,1)=FLOAT(NCT34)
HIST(36,1)=FLOAT(NCT35)
HIST(37,1)=FLOAT(NCT36)
HIST(38,1)=FLOAT(NCT37)
HIST(39,1)=FLOAT(NCT38)
HIST(40,1)=FLOAT(NCT39)
HIST(41,1)=FLOAT(NCT40)
HIST(42,1)=FLOAT(NCT41)
HIST(43,1)=FLOAT(NCT42)
HIST(44,1)=FLOAT(NCT43)
HIST(45,1)=FLOAT(NCT44)
HIST(46,1)=FLOAT(NCT45)
HIST(47,1)=FLOAT(NCT46)
HIST(48,1)=FLOAT(NCT47)
HIST(49,1)=FLOAT(NCT48)
HIST(50,1)=FLOAT(NCT49)
HIST(51,1)=FLOAT(NCT50)
HIST(52,1)=FLOAT(NCT51)
HIST(53,1)=FLOAT(NCT52)

OPEN(31,FILE='HIST.DAT',STATUS='NEW',FORM='FORMATTED')
DO I=1,53
   WRITE(31,*)HIST(I,1)
END DO
CLOSE(31)
TOP=0.
DO I=1,53
   DIFF1=TOP-HIST(I,1)
   IF(DIFF1.LT.0) THEN
      TOP=HIST(I,1)
   ENDIF
END DO

DO I=1,53
   DIFF2=TOP-HIST(I,1)
   IF(DIFF2.EQ.0) THEN
     NLO=I
     NLOB=I
```

```
          END IF
        END DO




        DO I=NLO,53
          DIFF3=HIST(I,1)-TOP1
          IF(DIFF3.LE.0)THEN
          TOP1=HIST(I,1)
          GOTO 29
          END IF
        END DO

29      DO I=NLO,53
          DIFF4=TOP1-HIST(I,1)
          IF(DIFF4.EQ.0) THEN
            NHIB=I
            NHI=I
          END IF
        END DO


        DO J=1,53
          DIFF5=HIST(J,1)-1000.
          IF(DIFF5.LT.0) THEN
            GOTO 31
          END IF
        END DO
31      NLOJ=J

        IF(NLOJ.LT.NLO) THEN
          NLOB=NLOJ
          NLO=NLOJ
        END IF

        IF(NLO.EQ.1) THEN
          NLO=0
        ELSEIF(NLO.EQ.2) THEN
          NLO=1
        ELSEIF(NLO.EQ.3) THEN
          NLO=5
        ELSEIF(NLO.EQ.4) THEN
          NLO=10
        ELSEIF(NLO.EQ.5) THEN
          NLO=15
        ELSEIF(NLO.EQ.6) THEN
          NLO=20
        ELSEIF(NLO.EQ.7) THEN
          NLO=25
        ELSEIF(NLO.EQ.8) THEN
          NLO=30
        ELSEIF(NLO.EQ.9) THEN
          NLO=35
        ELSEIF(NLO.EQ.10) THEN
          NLO=40
        ELSEIF(NLO.EQ.11) THEN
          NLO=45
```

```
ELSEIF(NLO.EQ.12) THEN
   NLO=50
ELSEIF(NLO.EQ.13) THEN
   NLO=55
ELSEIF(NLO.EQ.14) THEN
   NLO=60
ELSEIF(NLO.EQ.15) THEN
   NLO=65
ELSEIF(NLO.EQ.16) THEN
   NLO=70
ELSEIF(NLO.EQ.17) THEN
   NLO=75
ELSEIF(NLO.EQ.18) THEN
   NLO=80
ELSEIF(NLO.EQ.19) THEN
   NLO=85
ELSEIF(NLO.EQ.20) THEN
   NLO=90
ELSEIF(NLO.EQ.21) THEN
   NLO=95
ELSEIF(NLO.EQ.22) THEN
   NLO=100
ELSEIF(NLO.EQ.23) THEN
   NLO=105
ELSEIF(NLO.EQ.24) THEN
   NLO=110
ELSEIF(NLO.EQ.25) THEN
   NLO=115
ELSEIF(NLO.EQ.26) THEN
   NLO=120
ELSEIF(NLO.EQ.27) THEN
   NLO=125
ELSEIF(NLO.EQ.28) THEN
   NLO=130
ELSEIF(NLO.EQ.29) THEN
   NLO=135
ELSEIF(NLO.EQ.30) THEN
   NLO=140
ELSEIF(NLO.EQ.31) THEN
   NLO=145
ELSEIF(NLO.EQ.32) THEN
   NLO=150
ELSEIF(NLO.EQ.33) THEN
   NLO=155
ELSEIF(NLO.EQ.34) THEN
   NLO=160
ELSEIF(NLO.EQ.35) THEN
   NLO=165
ELSEIF(NLO.EQ.36) THEN
   NLO=170
ELSEIF(NLO.EQ.37) THEN
   NLO=175
ELSEIF(NLO.EQ.38) THEN
   NLO=180
ELSEIF(NLO.EQ.39) THEN
   NLO=185
ELSEIF(NLO.EQ.40) THEN
   NLO=190
ELSEIF(NLO.EQ.41) THEN
   NLO=195
```

```
ELSEIF(NLO.EQ.42) THEN
   NLO=200
ELSEIF(NLO.EQ.43) THEN
   NLO=205
ELSEIF(NLO.EQ.44) THEN
   NLO=210
ELSEIF(NLO.EQ.45) THEN
   NLO=215
ELSEIF(NLO.EQ.46) THEN
   NLO=220
ELSEIF(NLO.EQ.47) THEN
   NLO=225
ELSEIF(NLO.EQ.48) THEN
   NLO=230
ELSEIF(NLO.EQ.49) THEN
   NLO=235
ELSEIF(NLO.EQ.50) THEN
   NLO=240
ELSEIF(NLO.EQ.51) THEN
   NLO=245
ELSEIF(NLO.EQ.52) THEN
   NLO=250
ELSE
   NLO=255
END IF

IF(NHI.EQ.1) THEN
   NHI=0
ELSEIF(NHI.EQ.2) THEN
   NHI=1
ELSEIF(NHI.EQ.3) THEN
   NHI=5
ELSEIF(NHI.EQ.4) THEN
   NHI=10
ELSEIF(NHI.EQ.5) THEN
   NHI=15
ELSEIF(NHI.EQ.6) THEN
   NHI=20
ELSEIF(NHI.EQ.7) THEN
   NHI=25
ELSEIF(NHI.EQ.8) THEN
   NHI=30
ELSEIF(NHI.EQ.9) THEN
   NHI=35
ELSEIF(NHI.EQ.10) THEN
   NHI=40
ELSEIF(NHI.EQ.11) THEN
   NHI=45
ELSEIF(NHI.EQ.12) THEN
   NHI=50
ELSEIF(NHI.EQ.13) THEN
   NHI=55
ELSEIF(NHI.EQ.14) THEN
   NHI=60
ELSEIF(NHI.EQ.15) THEN
   NHI=65
ELSEIF(NHI.EQ.16) THEN
   NHI=70
ELSEIF(NHI.EQ.17) THEN
   NHI=75
```

```
ELSEIF(NHI.EQ.18) THEN
   NHI=80
ELSEIF(NHI.EQ.19) THEN
   NHI=85
ELSEIF(NHI.EQ.20) THEN
   NHI=90
ELSEIF(NHI.EQ.21) THEN
   NHI=95
ELSEIF(NHI.EQ.22) THEN
   NHI=100
ELSEIF(NHI.EQ.23) THEN
   NHI=105
ELSEIF(NHI.EQ.24) THEN
   NHI=110
ELSEIF(NHI.EQ.25) THEN
   NHI=115
ELSEIF(NHI.EQ.26) THEN
   NHI=120
ELSEIF(NHI.EQ.27) THEN
   NHI=125
ELSEIF(NHI.EQ.28) THEN
   NHI=130
ELSEIF(NHI.EQ.29) THEN
   NHI=135
ELSEIF(N.EQ.30) THEN
   NHI=140
ELSEIF(NHI.EQ.31) THEN
   NHI=145
ELSEIF(NHI.EQ.32) THEN
   NHI=150
ELSEIF(NHI.EQ.33) THEN
   NHI=155
ELSEIF(NHI.EQ.34) THEN
   NHI=160
ELSEIF(NHI.EQ.35) THEN
   NHI=165
ELSEIF(NHI.EQ.36) THEN
   NHI=170
ELSEIF(NHI.EQ.37) THEN
   NHI=175
ELSEIF(NHI.EQ.38) THEN
   NHI=180
ELSEIF(NHI.EQ.39) THEN
   NHI=185
ELSEIF(NHI.EQ.40) THEN
   NHI=190
ELSEIF(NHI.EQ.41) THEN
   NHI=195
ELSEIF(NHI.EQ.42) THEN
   NHI=200
ELSEIF(NHI.EQ.43) THEN
   NHI=205
ELSEIF(NHI.EQ.44) THEN
   NHI=210
ELSEIF(NHI.EQ.45) THEN
   NHI=215
ELSEIF(NHI.EQ.46) THEN
   NHI=220
ELSEIF(NHI.EQ.47) THEN
   NHI=225
```

```
      ELSEIF(NHI.EQ.48) THEN
         NHI=230
      ELSEIF(NHI.EQ.49) THEN
         NHI=235
      ELSEIF(NHI.EQ.50) THEN
         NHI=240
      ELSEIF(NHI.EQ.51) THEN
         NHI=245
      ELSEIF(NHI.EQ.52) THEN
         NHI=250
      ELSE
         NHI=255
      END IF


      DO I=1,53
         IF(HIST(I,1).LT.1)THEN
            HIST(I,1)=1.
         END IF
         RLHIST(I,1)=LOG10(HIST(I,1))
         IF(HIST(I,1).EQ.1)THEN
            HIST(I,1)=0.
         END IF
      END DO

      CALL MATSAV(1,MATRIX,53,53,1,0,
     +  HIST,DUMMY,'(1P2E25.17)')
      CALL MATSAV(1,LOGMAT,53,53,1,0,
     +  RLHIST,DUMMY,'(1P2E25.17)')

      DO I=1,53
         HIST(I,1)=0.
      END DO

      RETURN
      END
```

```
      SUBROUTINE MATSAV ( LUNIT, NAME, NR, M, N, IMG,
     $                    XREAL, XIMAG, FORMT )
C
C ------------------------------------------------------------------
C |            MATRIXx TM Software V7.0    (C) Copyright 1988
C |          INTEGRATED SYSTEMS INC., Santa Clara, California
C |
C | Published Work.  Permission is granted to any individual or
C | institution to modify, copy or use this subroutine except for
C | explicitly commercial purposes.
C ------------------------------------------------------------------
C |
C | MATSAV writes a matrix to a file in a MATRIXx-readable format.  It
C | can be called by programs that want to write data to MATRIXx.
C |
C | Files written with MATSAV can be read into MATRIXx with the LOAD
C | command.
C |
C |-----------------------------------------------------------------
C | Param. | Type              | On input-                        | On output-
C |-----------------------------------------------------------------
C |        |                   |                                  |
C | LUNIT  | INTEGER           | Fortran logical unit number.     | unchanged.
C |        |                   |                                  |
C | NAME   | CHARACTER*(*)     | Name of the matrix.  One al-     | unchanged.
C |        | (maximum          | phabetic followed by up to 9     |
C |        |  length 10)       | alphanumeric characters.         |
C |        |                   |                                  |
C | NR     | INTEGER           | Row-dimension in the             | unchanged.
C |        |                   | defining dimension or type       |
C |        |                   | statement in the calling         |
C |        |                   | program. NR must be greater      |
C |        |                   | than or equal to M.              |
C |        |                   |                                  |
C | M      | INTEGER           | Number of rows of the matrix     | unchanged.
C |        |                   |                                  |
C | N      | INTEGER           | Number of columns of the         | unchanged.
C |        |                   | matrix.                          |
C |        |                   |                                  |
C | IMG    | INTEGER           | If IMG = 0, the imaginary        | unchanged.
C |        |                   | part (XIMAG) is assumed to       |
C |        |                   | be zero and is not saved.        |
C |        |                   |                                  |
C | XREAL  | DOUBLE            | Real part of the matrix to       | unchanged.
C |        | PRECISION         | be saved.                        |
C |        |                   |                                  |
C | XIMAG  | DOUBLE            | Imaginary part of the matrix     | unchanged.
C |        | PRECISION         | to be saved.                     |
C |        |                   |                                  |
C | FORMT  | CHARACTER*(*)     | String containing the For-       | unchanged.
C |        | (maximum          | tran format to be used for       |
C |        |  length 20)       | writing the elements of the      |
C |        |                   | matrix.                          |
C |        |                   |                                  |
C |        |                   | For machine-independent files    |
C |        |                   | use '(1P2E25.17)'                |
C |        |                   |                                  |
C |        |                   | For machine-dependent (fast,     |
C |        |                   | compact) files use '(10A8)'      |
C |        |                   | or '(10Z16)'                     |
```

```
C  ------------------------------------------------------------------
C  |                                                                |
C  |  Example:  The following Fortran program generates a simple    |
C  |            identity matrix in X and writes it to Fortran unit 1.|
C  |            Assume that unit 1 has been preallocated as file (data|
C  |            set) TEST.                                          |
C  |                                                                |
C  |                 DOUBLE PRECISION X(20,3), DUMMY                |
C  |                   DO 200 J=1,3                                 |
C  |                     DO 100 I=1,10                              |
C  |                       X(I,J)=0.0D0                             |
C  |              100      CONTINUE                                 |
C  |                       X(J,J)=1.0D0                             |
C  |              200    CONTINUE                                   |
C  |                                                                |
C  |                    CALL MATSAV( 1, 'AMATRIX', 20, 10, 3, 0,    |
C  |                   $             X, DUMMY, '(1P2E25.17)' )      |
C  |                                                                |
C  |                    STOP                                        |
C  |                    END                                         |
C  |                                                                |
C  |                                                                |
C  |  After this program runs, invoke MATRIXx and type:             |
C  |                                                                |
C  |                 <> LOAD 'TEST'                                 |
C  |                                                                |
C  |  This will put X on the stack as a variable named AMATRIX.     |
C  |                                                                |
C  |  When the objective is to convert a file written by another soft-|
C  |  ware package into a MATRIXx-readable file, you will have to write|
C  |  a small program that reads the data in using the format of the |
C  |  other software package then writes it out by calling MATSAV.   |
C  ------------------------------------------------------------------
C
      INTEGER             LUNIT, M, N, NR, IMG
      CHARACTER*(*)       NAME, FORMT
      DIMENSION           XREAL(NR,*), XIMAG(NR,*)
      CHARACTER           NAM*10, FORM*20
C
C                               ------------------------------------
C                               | write header record.             |
C                               ------------------------------------

      NAM=NAME
      FORM=FORMT
      OPEN(LUNIT,FILE=NAM,STATUS='NEW')

      WRITE(LUNIT,'(A10,3I5,A20)') NAM,M,N,IMG,FORM
C                               ------------------------------------
C                               | write real-part of the matrix.   |
C                               ------------------------------------
      WRITE(LUNIT,FORM) ((XREAL(I,J),I=1,M),J=1,N)
C                               ------------------------------------
C                               | write imaginary-part if nonzero. |
C                               ------------------------------------
      IF(IMG.NE.0) WRITE(LUNIT,FORM) ((XIMAG(I,J),I=1,M),J=1,N)



      CLOSE(LUNIT)
```

```
      RETURN
      END




C-----------------------------------------------------------------------C
C
C       SAME GET IMAGE SUBROUTINE FOUND IN BSG.FOR
C
C-----------------------------------------------------------------------C

      SUBROUTINE GET_FLIR_IM(NAME,AR,ROW_NUM,COL_NUM)
        IMPLICIT INTEGER(A-Z)
        BYTE             AR_BYTE(240,640)
        DIMENSION        AR(240,640)
        CHARACTER*24     NAME

        OPEN(10,FILE=NAME,STATUS='OLD',FORM='UNFORMATTED',
     +       RECORDTYPE='VARIABLE',ERR=20)
 20     OPEN(10,FILE=NAME,STATUS='OLD',FORM='UNFORMATTED',
     +       RECORDTYPE='FIXED')


        DO 100 R = 1,ROW_NUM
               READ(10) (AR_BYTE(R,C),C=1,COL_NUM)
 100    CONTINUE


        DO R = 1,ROW_NUM
               DO C = 1,COL_NUM
                      IF(AR_BYTE(R,C).LT.0)THEN
                      AR(R,C) = AR_BYTE(R,C)+256
                      ELSE
                      AR(R,C) = AR_BYTE(R,C)
                      END IF
               END DO
        END DO
        CLOSE(10)
        RETURN
        END



C-----------------------------------------------------------------------C
C
C       SAME SUBROUTINE FOUND IN BSG.FOR
C
C-----------------------------------------------------------------------C




      SUBROUTINE PUT_FLIR_IM(NAMEOUT,AR,ROW_NUM,COL_NUM)
        IMPLICIT INTEGER (A-Z)
```

```
      DIMENSION         AR(240,640)
      BYTE              AR_BYTE(240,640)
      CHARACTER*24      NAMEOUT
      DO R=1,ROW_NUM
              DO C=1,COL_NUM
                      AR_BYTE(R,C)=MOD(AR(R,C),256)-128
              END DO
      END DO
      OPEN(20,FILE=NAMEOUT,STATUS='NEW',FORM='UNFORMATTED',
     +     RECL=COL_NUM/4+1,RECORDTYPE='FIXED')
      DO 100 R=1,ROW_NUM
              WRITE(20) (AR_BYTE(R,C),C=1,COL_NUM)
100   CONTINUE
      CLOSE(20)
      RETURN
      END
```

Appendix E. Regions of interest FORTRAN source code.

SHOW_LAB_FLIR_IM.FOR

SYNOPSIS: The SHOW_LAB_FLIR_IM program uses UIS system library routines for graphic screen displays. This program determines the center pixel of every "blob" that passed through the THRESH_ADD_HIST program. From these on center pixels, a table of locations is generated. The total number of on center pixels and their respective locations are used to determine the number of jets and their location for generation.

INPUT: Thresholded Gabor filtered image.

OUTPUT: To the screen, a labeled version of the thresholded image.
The total number of locations of on center pixels.
The locations of the on center pixels.

```
      PROGRAM show_lab_flir_im

      IMPLICIT INTEGER (A-Z)
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'

      INTEGER*4        F_CR(255), F_CC(255), ROW_MAX(255)
      INTEGER*4        ROW_MIN(255), COL_MAX(255), COL_MIN(255)
      BYTE             F_IM(640,240)
      INTEGER*4        F_HOLD(640,240), WIN_NUM(200), TFLIR(640,240)
      INTEGER*4        TF1(240,640), TF2(240,640)
      CHARACTER*1      CONT, ER
      CHARACTER*40     N1, N2, OUT_NAME
      REAL             INTEN

      TYPE*,'PROGRAM TO ASSIST ALIGNMENT OF FLIR AND LR BLOBS'
      TYPE*,' '
      TYPE*,'WHEN PROMPTED, ENTER THE NAMES OF'
      TYPE*,'PREVIOUSLY SEGMENTED FLIR IMAGES'
      TYPE*,' '

      VCM_ID = UIS$CREATE_COLOR_MAP(256)
      VD_ID = UIS$CREATE_DISPLAY(0.0, 0.0, 640.0, 240.0, 20.0, 20.0,
     1                           VCM_ID)

      DO 10 I=1,250
         INTEN = I/256.0
         CALL UIS$SET_INTENSITY(VD_ID,(I-1),INTEN)
10    CONTINUE

      call uis$set_color(vd_id,250, 1.0, 1.0, 1.0)
      call uis$set_color(vd_id,251, 0.0, 0.0, 0.0)
      call uis$set_color(vd_id,252, 0.5, 0.5, 0.5)
      call uis$set_color(vd_id,253, 1.0, 1.0, 1.0)
      call uis$set_color(vd_id,254, 1.0, 1.0, 1.0)
      call uis$set_color(vd_id,255, 0.0, 0.0, 0.0)

      LAST=1
      WIN_COUNT=0
      CONT='Y'

      DO 5000 WHILE ((CONT.EQ.'Y').OR.(CONT.EQ.'y'))

         type*,'ENTER FILENAME OF FLIR IMAGE:'
         accept 50, n1
50       format(a)

         type*,'SEARCHING FOR FLIR FILE'
         open(unit=10, file=n1, status='old', form='unformatted',
     1        recordtype='variable', err=90)
90       open(unit=10, file=n1, status='old', form='unformatted',
     1        recordtype='fixed')
         type*,'FOUND FLIR FILE'
         rewind (10)  !MANUALLY SET POINTER TO FILE AT TOP OF FILE

C        READ THE FLIR IMAGE
         DO 130 J=1,240
            READ(10) (F_IM(I,J), I=1,640)
130      CONTINUE
         CLOSE(10)
```

```
C           CONVERT FLIR IMAGE TO INTEGER FORMAT
            DO 170 I=1,640
               DO 190 J=1,240
                  IF (F_IM(I,J).LT.0) THEN
                      TFLIR(I,J) = F_IM(I,J) + 256
                  ELSE
                      TFLIR(I,J) = F_IM(I,J)
                  END IF

                  TF1(J,I) = TFLIR(I,J)   !ROTATE FOR PROPER LABELLING OF
                                          !SEGMENTED IMAGE
 190           CONTINUE
 170        CONTINUE

C           LABEL REGIONS IN THE FLIR IMAGE
            CALL LABEL_REG(TF1, 240, 640, F_MAX, TF2)

C           ROTATE BACK FOR DISPLAY PURPOSES
            DO I=1,240
               DO J=1,640
                  F_HOLD(J,I) = TF2(I,J)
               END DO
            END DO
            write(*,101) f_max
 101        format(' NUMBER OF FLIR REGIONS DETECTED = ',I4)

            OPEN(47,FILE='TOT.DAT',STATUS='NEW',FORM='FORMATTED')
            WRITE(47,*)f_max
            CLOSE(47)



C           INITIALIZE HOLDING VARIABLES
            DO I=1,F_MAX
               COL_MAX(I) = 0
               COL_MIN(I) = 641
               ROW_MAX(I) = 0
               ROW_MIN(I) = 241
            END DO
C           FIND MAX AND MIN EXTENT OF EACH REGION
            DO I=1,640
               DO J=1,240
                  IF (F_HOLD(I,J).NE.0) THEN
                     TREG = F_HOLD(I,J)
                     IF (I.GT.COL_MAX(TREG)) THEN
                        COL_MAX(TREG) = I
                     END IF
                     IF (I.LT.COL_MIN(TREG)) THEN
                        COL_MIN(TREG) = I
                     END IF
                     IF (J.GT.ROW_MAX(TREG)) THEN
                        ROW_MAX(TREG) = J
                     END IF
                     IF (J.LT.ROW_MIN(TREG)) THEN
                        ROW_MIN(TREG) = J
                     END IF
                  END IF
               END DO
            END DO
```

```
            DO I=1,F_MAX
               F_CR(I) = (ROW_MAX(I) + ROW_MIN(I))/2
               F_CC(I) = (COL_MAX(I) + COL_MIN(I))/2
            END DO

            TYPE*,'LOCATION TABLE FOR CENTER PIXELS IN FLIR IMAGE'
            TYPE*,'REGION     ROW       COL'

            OPEN(47,FILE='LFP.DAT',STATUS='NEW',FORM='FORMATTED')

            DO 111 K=1,F_MAX
               WRITE (*,116) K, F_CR(K), F_CC(K)
               WRITE(47,*)F_CR(K),F_CC(K)
 116           FORMAT(I4,'    ',I4,'    ',I4)
 111        CONTINUE
            CLOSE(47)
C           PREPARE FLIR IMAGE FOR DISPLAY
            DO 410 I=1,640
               DO 430 J=1,240
                  IF (F_HOLD(I,J).NE.0) THEN
                     F_IM(I,J) = -126
                  ELSE
                     F_IM(I,J) = 0
                  END IF
 430              CONTINUE
 410        CONTINUE

C           TURN ON PIXELS DETERMINED TO BE CENTER PIXELS
            DO 412 I=1,F_MAX
               F_IM(F_CC(I),F_CR(I)) = -7 !-7 IN 2'S COMP = 249 IN UNSIGNED LINGO
 412        CONTINUE

            CALL DISPLAY_DRIVER1(VD_ID, F_IM, 240, 640, 15, 17, N1,
     1                           22.0, 9.0, 640.0, 240.0, WD_ID, F_CR,
     1                           F_CC, F_MAX)
            WIN_COUNT=WIN_COUNT + 1
            WIN_NUM(WIN_COUNT) = WD_ID

            write(*,690)
 690        format(' CONTINUE? (Y/N) > ',$)
            accept 710, cont
 710        format(a1)


            write (*,730)
 730        format(' ERASE CURRENT IMAGES? (Y/N) > ',$)
            accept 750, er
 750        format(a1)

            if ((er.eq.'Y').or.(er.eq.'y')) then
               DO 770 I=LAST,WIN_COUNT
                  CALL UIS$DELETE_WINDOW(WIN_NUM(I))
 770              CONTINUE
               LAST=WIN_COUNT + 1
            END IF

 5000    CONTINUE

         END
```

```
C-----------------------------------------------------------------------
      SUBROUTINE LABEL_REG(IM_IN, ROW, COL, NUM_REG, IM_OUT)

      IMPLICIT INTEGER(A-Z)

      DIMENSION       IM_IN(ROW,COL), IM_OUT(ROW,COL)
      INTEGER         STAT(640,640)
      INTEGER         NN_ROW(256*640), NN_COL(256*640)
      INTEGER         FLAG(8)

C     INITIALIZE NEEDED VARIABLES
      DO R=1,ROW
         DO C=1,COL
            IM_OUT(R,C) = 0
            STAT(R,C) = 0
            !SET  OUTER ROWS,COLS EQUAL TO ZERO
            IF ((R.EQ.1).OR.(R.EQ.ROW).OR.
     1           (C.EQ.1).OR.(C.EQ.COL))     THEN
               IM_IN(R,C) = 0
            END IF
         END DO
      END DO
      DO I=1,ROW*COL
         NN_ROW(I) = 0
         NN_COL(I) = 0
      END DO
      REG_CNT = 0

C     BEGIN REGION LABELLING SCHEME
      DO R=2,(ROW-1)
         DO C=2,(COL-1)

            !WHEN NONZERO PIXEL IS FOUND, BEGIN
            IF ((IM_IN(R,C).NE.0).AND.(STAT(R,C).EQ.0)) THEN

               LEN = 0
               IND = 0
               REG_CNT = REG_CNT + 1
               IM_OUT(R,C) = REG_CNT

               !SEARCH NEAREST NEIGHBORHOOD OF NONZERO PIXEL
               K = 0
               DO I = -1,1
                  DO J = -1,1

                     K=K+1
                     !IF NEAREST NEIGHBORS ARE FOUND ON, THEN
                     !SEED NEAREST NEIGHBOR TABLE
                     IF ((IM_IN(R+I,C+J).NE.0).AND.
     1                    (STAT(R+I,C+J).EQ.0))       THEN
                        LEN = 1
                        NN_ROW(LEN) = R+I
                        NN_COL(LEN) = C+J
                        FLAG(K) = 1
                        IM_OUT(R+I,C+J) = REG_CNT
                     ELSE
                        FLAG(K) = 0
                     END IF
                     STAT(R+I,C+J) = 1
```

```
                    END DO
                 END DO

C                EXAMINE NEAREST NEIGHBOR TABLE AND APPEND OTHER NEAREST
C                NEIGHBORS FOUND
                 F1 = 0
                 DO I=1,8
                    IF (FLAG(I).NE.0) THEN
                       F1 = 1
                    END IF
                 END DO

                 IF (F1.NE.0) THEN
                    DO WHILE ((IND.LT.LEN).OR.(F1.EQ.1))

                       DO I=1,8
                          FLAG(I) = 0
                       END DO
                       IND = IND + 1

                       TR = NN_ROW(IND)
                       TC = NN_COL(IND)
                       K=0
                       DO I=-1,1
                          DO J=-1,1
                             K=K+1
                             IF ((IM_IN(TR+I,TC+J).NE.0).AND.
     1                          (STAT(TR+I,TC+J).EQ.0))     THEN
                                LEN=LEN+1
                                NN_ROW(LEN) = TR+I
                                NN_COL(LEN) = TC+J
                                IM_OUT(TR+I,TC+J) = REG_CNT
                                FLAG(K) = 1
                             END IF
                             STAT(TR+I,TC+J) = 1
                          END DO
                       END DO
                       F1 = 0
                       DO I=1,8
                          IF (FLAG(I).NE.0) THEN
                             F1 = 1
                          END IF
                       END DO

                    END DO

                 END IF

              ELSE

                 STAT(R,C) = 1

              END IF

           END DO
        END DO

        NUM_REG = REG_CNT

        RETURN
```

```
      END
C-----------------------------------------------------------------------
C   A SUBROUTINE TO DRIVE THE VAX STATION II DISPLAY FOR RASTER IMAGES.
C
C   INPUTS ARE THE VIRTUAL COLOR MAP ID, VCM_ID, THE IMAGE ARRAY IN BYTE
C   FORMAT, NUMBER OF ROWS AND COLUMNS, X,Y CDS OF LOWER LEFT CORNER
C   OF RASTER IMAGE AS IT WILL APPEAR IN THE DISPLAY WINDOW,
C   A NAME TO BE PLACED IN THE DISPLAY BANNER, THE X AND Y EXTENT
C   OF THE WINDOW FOR DISPLAYING THE IMAGE, AND THE SIZE IN X AND Y
C   DIMENSIONS OF THE VIRTUAL DISPLAY WINDOW OPPENED IN THE MAIN.
C
C   OUTPUT IS THE DISPLAY OF THE IMAGE AND THE WINDOW ID NUMBER, WD_ID,
C   OF THE CURRENT WINDOW OPEN.

      SUBROUTINE DISPLAY_DRIVER1 (VD_ID, IMAGE, ROW_NUM, COL_NUM,
     1                            X_LOWER_LEFT, Y_LOWER_LEFT, NAME,
     1                            X_SIZE, Y_SIZE, VD_X_SIZE,
     1                            VD_Y_SIZE, WD_ID, ROW_TABLE,
     1                            COL_TABLE, NUM_EL)

      IMPLICIT INTEGER (A-Z)
      CHARACTER*4   CHAR
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'
      REAL  X_SIZE, Y_SIZE, VD_X_SIZE, VD_Y_SIZE
      DIMENSION    IMAGE(COL_NUM, ROW_NUM) ! IMAGE IS PASSED IN IN BYTE TYPE
      DIMENSION    COL_TABLE(NUM_EL)
      DIMENSION    ROW_TABLE(NUM_EL)
      character*40 out_name

      WD_ID = UIS$CREATE_WINDOW(VD_ID,'SYS$WORKSTATION',NAME,
     1                          0.0,0.0,VD_X_SIZE, VD_Y_SIZE,
     1                          X_SIZE, Y_SIZE)

      CALL UIS$SET_WRITING_MODE(VD_ID, 0,1,UIS$C_MODE_COPY)

      CALL UISDC$IMAGE(WD_ID,1,X_LOWER_LEFT,Y_LOWER_LEFT,
     1                 X_LOWER_LEFT + COL_NUM,
     1                 Y_LOWER_LEFT + ROW_NUM,
     1                 COL_NUM, ROW_NUM, 8, IMAGE)

      CALL UIS$SET_WRITING_INDEX(VD_ID, 0, 2, 249)

      CALL UIS$SET_FONT(VD_ID, 2, 2,
     1                  'DTABER0I03WK00GG0001UZZZZ02A000')

      CALL UIS$SET_WRITING_MODE(VD_ID, 2, 2, UIS$C_MODE_OVER)

      DO 100 I=1,NUM_EL
         X=col_table(I) - 3  + X_LOWER_LEFT
         Y=ROW_NUM - row_table(I) + 2 + Y_LOWER_LEFT
         ENCODE(4, 2000,CHAR) I
2000     FORMAT(I4)
         CALL UISDC$TEXT(WD_ID, 2, CHAR, x, y )

100   CONTINUE

      CALL UISDC$TEXT(WD_ID, 2, 'ORIG', 10,10)
```

```
c  This section changed by K Ayer for saving file for output to laser
c  printer

          type*,'What do you wish to call the output image file?'
          accept 50, out_name
          call put_flir_im(out_name,image,240,640)

50        format(a)

          RETURN
          END



       SUBROUTINE PUT_FLIR_IM(NAMEOUT,AR,ROW_NUM,COL_NUM)
          IMPLICIT INTEGER (A-Z)
          DIMENSION        AR(240,640)
          BYTE             AR_BYTE(240,640)
          CHARACTER*24     NAMEOUT
          DO R=1,ROW_NUM
                  DO C=1,COL_NUM
                     ar_byte(r,c)=mod(ar(r,c),256)-128
                  END DO
          END DO
          OPEN(20,FILE=NAMEOUT,STATUS='NEW',FORM='UNFORMATTED',
      +        RECL=COL_NUM/4+1,RECORDTYPE='FIXED')
          DO 100 R=1,ROW_NUM
                  WRITE(20) (AR_BYTE(R,C),C=1,COL_NUM)
 100      CONTINUE
          CLOSE(20)
          RETURN
          END
```

Appendix F. Image display FORTRAN source code.

DISPLAY_IMAGES.FOR

SYNOPSIS: DISPLAY_IMAGES program displays FLIR and LADAR images based on user defined inputs. This program converts a two's complement BYTE stored image into proper format for UIS display routines. Most of this program is for LADAR image display and is not pertinent to this research effort. However, those portions for FLIR image display take a 240 by 640 pixel image, convert it into UIS format for screen display. Color maps, mouse commands, and other UIS display routines may be found in the UIS SYS$LIBRARY graphics packages readily accessible on most MicroVAX Workstations.

Image Display.

Hardware. Images generated by the FORTRAN codes found in Appendix C, D, and E were displayed using a Digital Corporation VaxStation 3 mini mainframe computer with a Digital Corporation 19 inch multisync, multicolor cathode ray tube (CRT).

Software. The FORTRAN code required to display the images on screen was provided by Capt Michael Roggemann. This code was not intended for general purposes, but rather for the specific use of displaying FLIR and Laser Radar (LADAR) images.

This display program incorporated the UIS (VAX graphics display language) display routines specific to VMS MicroVax workstations, and therefore, was not portable to other operating systems. This statement should not be construed as a limitation on the Gabor transform to segment images but rather a limitation on the display of the Gabor filtered images.

INPUT: FLIR or LADAR image(s).

OUTPUT: Screen display of FLIR or LADAR image(s).

```
C    A MAIN PROGRAM TO DISPLAY FLIR AND LASER RADAR IMAGES ON A MICROVAX
C    WORKSTATION DISPLAY USING UIS DISPLAY ROUTINES.
C
C    THIS PROGRAM IS NOT A GENERAL DISPLAY PROGRAM.  IT IS SPECIFICALLY WRITTEN
C    TO DISPLAY FLIR IMAGES STORED IN BYTE FORMAT (1 BYTE/PIXEL,
C    AND LASER RADAR  RESOLVED RANGE IMAGES STORED IN
C    INTEGER FORMAT (4 BYTE/PIXEL).
C
C    USER OPTIONS ARE AVAILABLE TO SELECT THE FOLLOWING:
C           - TYPE OF IMAGE TO BE DISPLAYED; FLIR OR LASER RADAR RESOLVED RANGE
C           - WHETHER OR NOT THE IMAGE IS TO BE VIEWED IN COLOR
C           - SIZE OF LASER RADAR IMAGE. THREE SIZES SUPPORTED:
C               -- 127 COL BY 64 ROW
C               -- 255 COL BY 128 ROW
C               -- 511 COL BY 256 ROW
C           - WHETHER OR NOT RANGE UNITS ARE ADJUSTED BY A FACTOR OF 10 (DIVIDED)
C           - WHETHER OR NOT A MODULO 256 OPERATION IS APPLIED TO DISPLAY FULL
C             DYNAMIC RANGE OF RESOLVED RANGE IMAGE
C    ONLY ONE SIZE OF FLIR IMAGE IS CURRENTLY SUPPORTED: 640 COL BY 240 ROW.
C
C    PROGRAMMED BY:  CAPT MIKE ROGGEMANN, COMPLETETED 9 JAN 88
C-------------------------------------------------------------------------------

      program DISPLAY_IMAGES

      IMPLICIT INTEGER (A-Z)
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'

      INTEGER*4        RNG_IM(511,256)
      BYTE             FLIR_IM(256,256), DISPLAY_IM(256,256)
      BYTE             DISPLAY_IM1(127,64),DISPLAY_IM2(255,128)
      BYTE             DISPLAY_IM3(511,256)
      CHARACTER*40     NAME1
      CHARACTER*1      CONTINUE, ADJUST, MOD_256, DEL, CLR
      CHARACTER*1      CB
      REAL             TEMP, QUO, COLOR, INTEN

      INTEGER*4        WINDOW_NUM(200)
      EQUIVALENCE (RNG_IM,RNG_HOLD)

      type*, ' '
      type*,'DISPLAY PROGRAM FOR RESOLVED RANGE AND FLIR IMAGES'
      type*, ' '
      type*, ' '

C  CREATE A COLOR MAP

      !  VIRUTAL COLOR MAP ID NO. FOR GRAY SCALE DISPLAY
      VCM_ID1 = UIS$CREATE_COLOR_MAP(256)

      !  VIRTUAL COLOR MAP ID NO. FOR COLOR DISPLAY
      VCM_ID2 = UIS$CREATE_COLOR_MAP(256)

      ! CREATE DISPLAY FOR GRAY SCALE IMAGES
      VD_ID1 = UIS$CREATE_DISPLAY(0.0,0.0,256.0,256.0,20.0,
     1                            8.0,VCM_ID1)

      ! CREATE DISPLAY FOR COLOR IMAGES
      VD_ID2 = UIS$CREATE_DISPLAY(0.0,0.0,256.0,256.0,20.0,
```

```
     1                              8.0,VCM_ID2)

C    LOAD THE COLOR , OR GRAY SCALE MAP

         DO 10 i=1,256    !  FILLS GRAY SCALE MAP
            INTEN = i/256.0
            j=i-1
            CALL UIS$SET_INTENSITY(vd_id1,j,INTEN)
 10      CONTINUE


         CALL FILL_COLOR_MAP(VD_ID2) !  FILLS COLOR MAP

C   ITERATION CONTROL FOR DISPLAYING MULTIPLE IMAGES
C   INITIALIZE REQUIRED VARIABLES

         LAST = 1
         WINDOW_COUNT = 0
         CONTINUE = 'Y'
         DO 5000 WHILE ( (CONTINUE.EQ.'Y').OR.(CONTINUE.EQ.'y') )

C      GET FILENAME OF FILE TO BE DISPLAYED
            type*,'ENTER FILENAME OF IMAGE TO BE DISPLAYED'
            accept 50, name1
 50         format(a)

C    QUERY FOR REQUIRED USER INPUT

            type*,'IS THE FILE:  (1) RESOLVED RANGE LASER RADAR'
            type*,'              (2) FLIR'
            write(5,100)
 100        format(' ENTER 1 OR 2 >  ', $)
            accept*,image_type
            type*,' '

            write(*,335)
 335        format(' DRAW IN COLOR?(Y/N) > ',$)
            accept 340, clr
 340        format(a1)

            write (*,336)
 336        format( ' DRAW COLOR BAR?(Y/N) > ',$)
            accept 341, cb
 341        format(a1)

C    SET VD_ID NUMBER APPROPRIATE FOR EITHER COLOR OR GRAY SCALE DISPLAY
            if ((clr.eq.'Y').or.(clr.eq.'y')) then
               VD_ID = VD_ID2
            else
               VD_ID = VD_ID1
            end if

C    CONDITIONAL HANDLING FOR RESOLVED RANGE IMAGES
            if (image_type.eq.1) then


C    OPEN THE FILE
            open(unit=10,file=name1,status='old',form='unformatted',
     1         recordtype='variable', err=105)
```

```
C   OPEN STATEMENT TO HANDLE EXCEPTION THAT RECORDTYPE = FIXED
 105          open(unit=10,file=name1,status='old',form='unformatted',
     1           recordtype='fixed')
              rewind(10) !ASSURES POINTER SET TO TOP OF FILE

              type*,'FILE FOUND'
              type*, ' '

C             GET IMAGE SIZE INFORMATION FROM USER
              type*,'IS THE IMAGE  (1) 64 ROW BY 127 COL'
              type*,'              (2) 128 ROW BY 255 COL'
              type*,'              (3) 256 ROW BY 511 COL'
              write(5,200)
 200          format(' ENTER 1, 2, OR 3 > ', $)
              accept*,image_size
              type*,' '

C             QUERY FOR NEED TO ADJUST RANGE UNITS AND MODULO 256
              write(*,250)
 250          format(' ADJUST RANGE UNITS BY FACTOR OF 10?(Y/N) > ',$)
              accept 275, adjust
 275          format(a1)
              type*, ' '
              write(*,300)
 300          format(' MODULO 256 FOR DISPLAY?(Y/N) > ',$)
              accept 325, mod_256
 325          format(a1)
              type*,' '
              write(*,330)
 330          format(' DISPLAY RANGE SLICE? (Y/N) > ',$)
              accept 355, rng_slc
 355          format(a1)

C   GET MEDIAN RANGE REQUIRED FOR RANGE SLICE OPTION
              if ((rng_slc.eq.'Y').or.(rng_slc.eq.'y')) then
                 write(*,360)
 360             format(' ENTER MEDIAN RANGE OF RANGE SLICE IN UNITS',$,
     1                  ' OF THE IMAGE > ',$)
                 accept*, med_rng
              end if

C   SET IMAGE SIZE FOR LATER PROCESSING
              if (image_size.eq.1) then
                 row_num = 64
                 col_num = 127
              end if

              if (image_size.eq.2) then
                 row_num = 128
                 col_num = 255
              end if

              if (image_size.eq.3) then
                 row_num = 256
                 col_num = 511
              end if

C   READ THE IMAGE FILE INTO A TEMPORARY ARRAY
C   NOTE THAT THE METHOD OF READING RECORDS DIFFERS FROM ADA
C   THIS FORMAT FOR READING IMAGE FILES STORED IN ROW-MAJOR FORMAT
```

```
C    IS, HOWEVER, COMPATIBLE WITH THE UIS DRAWING ROUTINES, WHICH
C    ASSUME ARRAYS PASSED IN ARE IN X,Y FORMAT
               DO 350 j = 1,row_num
                  read(10) (rng_im(i,j), i = 1,col_num)
  350           CONTINUE

               type*,'FILE READING COMPLETE'

C    IF REQUIRED, ADJUST RANGE UNITS BY FACTOR OF 10
               if ((adjust.eq.'Y').or.(adjust.eq.'y')) then
                  DO 375 i = 1,col_num
                     DO 380 j = 1,row_num
                        rng_im(i,j) = rng_im(i,j)/10
 380                 CONTINUE
 375                 CONTINUE
                     type*,'RANGE UNITS ADJUSTED'
               end if

C    IF REQUIRED, PERFORM MODULO 256 OPERATION TO PREPARE FOR DISPLAY
               if ((mod_256.eq.'Y').or.(mod_256.eq.'y')) then
                  DO 385 i = 1,col_num
                     DO 390 j = 1,row_num
                        TEMPA = RNG_IM(i,j)
                        rng_im(i,j) = JMOD(TEMPA,256)
 390                 CONTINUE
 385                 CONTINUE
                     type*,'MODULO 256 COMPLETE'
               end if

C    IF REQUIRED, COMPUTE RANGE SLICE OF IMAGE
               if ((rng_slc.eq.'Y').or.(rng_slc.eq.'y')) then
                  CALL RANGE_SLICE (rng_im, 256, 511, med_rng)
                  type*,'RANGE SLICE COMPUTATION COMPLETE'
               end if

C    ELIMINATE PIXELS AT OR ABOVE 250, AS DISPLAY RESETS THESE TO MANAGE
C    DEFAULT CONTROL WINDOWS TO VARIOUS SHADES OF GRAY
               DO 401 i=1,col_num
                  DO 402 j=1,row_num
                     if (rng_im(i,j).ge.250) then
                        rng_im(i,j) = 249
                     end if
 402                 CONTINUE
 401           CONTINUE

C    CONVERT FROM INTEGER TO (SIGNED) BYTE FORMAT
               DO 410 i=1,col_num
                  DO 420 j=1,row_num

                     if (rng_im(i,j).le.127) then
                        rng_im(i,j)=rng_im(i,j)
                     else
                        rng_im(i,j)=rng_im(i,j)-256
                     end if

 420                 CONTINUE
 410           CONTINUE

C    NOW, ASSIGN ELEMENTS OF RNG_IM TO PROPERLY SIZED DISPLAY_IM ARRAY
C    AND CALL THE UIS DISPLAY ROUTINES
```

```
            if (image_size.eq.1) then  ! 127 col by 64 row array
               DO 800 i=1,col_num
                  DO 810 j=1,row_num
                     display_im1(i,j) = rng_im(i,j)
 810              CONTINUE
 800           CONTINUE

               type*,'DISPLAY IMAGE READY'

               CALL DISPLAY_DRIVER(VD_ID, DISPLAY_IM1, ROW_NUM,
     1                             COL_NUM, 28, 29, NAME1,
     1                             6.0, 4.0, 256.0, 256.0, WD_ID)

               WINDOW_COUNT = WINDOW_COUNT + 1
               WINDOW_NUM(WINDOW_COUNT) = WD_ID

            end if

            if (image_size.eq.2) then  ! 128 row by 255 col array
               DO 820 i=1,col_num
                  DO 830 j=1,row_num
                     display_im2(i,j) = rng_im(i,j)
 830              CONTINUE
 820           CONTINUE

               type*,'DISPLAY IMAGE READY'

               CALL DISPLAY_DRIVER (VD_ID, DISPLAY_IM2, ROW_NUM,
     1                              COL_NUM, 25, 28, NAME1, 10.0,
     1                              6.0, 256.0, 256.0, WD_ID)

               WINDOW_COUNT = WINDOW_COUNT + 1
               WINDOW_NUM(WINDOW_COUNT) = WD_ID

            end if

            if (image_size.eq.3) then  ! 256 row by 511 col array
               DO 840 i=1,col_num
                  DO 850 j=1,row_num
                     display_im3(i,j) = rng_im(i,j)
 850              CONTINUE
 840           CONTINUE

               type*,'DISPLAY IMAGE READY'

               CALL DISPLAY_DRIVER(VD_ID, DISPLAY_IM3, ROW_NUM,
     1                             COL_NUM, 19, 25, NAME1, 18.0,
     1                             10.0, 256.0, 256.0, WD_ID)

               WINDOW_COUNT = WINDOW_COUNT + 1
               WINDOW_NUM(WINDOW_COUNT) = WD_ID

            end if

         end if  ! END OF CONDITIONAL TREATMENT OF RESOLVED RANGE IMAGE

C   DISPLAY PREPARATION FOR FLIR IMAGES
         if (image_type.eq.2) then
            row_num = 256
            col_num = 256
```

```
C   OPEN THE FILE
            open(unit=10,file=name1,status='old',form='unformatted',
     1         recordtype='variable',err=110)

    EXECPTION HANDLER FOR CASE WHEN RECORDTYPE=FIXED
 10         open(unit=10,file=name1,status='old',form='unformatted',
     1         recordtype='fixed',access='sequential')
            rewind(10) !ASSURES POINTER SET TO TOP OF FILE

            type*,'FILE FOUND'

C   GET A FLIR IMAGE
            DO 600 j = 1,row_num
               read(10) (display_im(i,j), i = 1,col_num)
600         CONTINUE

C   SET ALL PIXELS AT OR ABOVE 250 TO 249
            DO 601 i=1,col_num
               DO 602 j=1,row_num
                  if ((display_im(i,j).le.-1).and.
     1                  (display_im(i,j).ge.-6)) then
                     display_im(i,j) = -7
                  end if
602            CONTINUE
601         CONTINUE

            type*,'DISPLAY IMAGE READY'

C   CALL REQUIRED UIS ROUTINES TO DISPLAY THE IMAGE

            CALL DISPLAY_DRIVER(VD_ID, DISPLAY_IM, ROW_NUM,
     1                          COL_NUM, 15, 17, NAME1, 22.0,
     1                          9.0, 256.0, 256.0, WD_ID)

            WINDOW_COUNT = WINDOW_COUNT + 1
            WINDOW_NUM(WINDOW_COUNT) = WD_ID

         end if  ! END OF CONDITIONAL TREATMENT FOR FLIR IMAGES

       close (10)

C   MAKE COLOR BAR IF DESIRED
       if ((cb.eq.'Y').or.(cb.eq.'y')) then
          CALL MAKE_COLOR_BAR
       end if

C   CONTINUE USER INTERFACE FOR DISPLAYING MULTIPLE IMAGES

       write(*,700)
700    format(' DISPLAY ANOTHER IMAGE?(Y/N) > ',$)
       accept 725, continue
725    format(a1)

       write(*,750)
750    format(' ERASE CURRENT IMAGES?(Y/N) > ',$)
       accept 775, del
775    format(a1)
```

```
      if ((del.eq.'Y').or.(del.eq.'y')) then

         DO 781 I=LAST,WINDOW_COUNT
            CALL UIS$DELETE_WINDOW(WINDOW_NUM(I))
781      CONTINUE

         LAST = WINDOW_COUNT + 1

      end if

5000  CONTINUE

C     DELETE VIRTUAL DISPLAY IN AN ATTEMPT TO RETURN SYSTEM TO PREVIOUS
C     SETTINGS, ETC
      CALL UIS$DELETE_DISPLAY(VD_ID)

      end
```

Appendix G. Jet generating FORTRAN source code.

JETS.FOR

SYNOPSIS: JETS.FOR program takes the locations of on center pixels generated by the SHOW_LAB_FLIR_IM program and builds jets at these locations. A jet is a "stack" of self similar Gabor correlation coefficients corresponding to the various Gabor sizes and orientations. For instance, a jet may consist of five Gabor sizes ranging from 128 by 128 to 8 by 8 with each subsequent layer decreasing in size by octaves, with four orientations per layer, thus creating a five layer, four sublayer stack of Gabor coefficients (20 coefficients associated with each location). The program generates an output image file for each location within the image. The first layer of the "jetted" image consists of four orientations (0, 45, 90, and 135 degrees) within a 128 by 128 Gabor window size. The second layer consist of four orientations within a Gabor window of 64 by 64. The third layer consists of four orientations within a 32 by 32 Gabor size window. The fourth layer consists of four orientations within a 16 by 16 Gabor window size. The five layer consists of four orientations within an eight by eight Gabor widow size.

INPUT: Self similar Gabor filtered FLIR image(s).
Total number of locations for jet construction.
Locations of the on center pixels for each "blob" in the image.

OUTPUT: Jet images at each on center pixel location.

```
C-------------------------------------------------------------------C
C
C       PROGRAM TO GENERATE JETS LOCATED AT POINTS FOUND BY THE
C       ROGGEMANN CENTROID FINDING ALGORITHM, SHOW_LAB_FLIR_IM
C
C       ARRAYS USED:
C
C               NAR             - INTEGER FORMAT OF INPUT IMAGE
C               NSUM            - INTEGER FORMAT OF THE SUPERIMPOSED
C                                 SS_GABORED IMAGES
C               NJET            - INTEGER FORMAT OF THE SUPERIMPOSED
C                                 JET COEFFICIENTS
C               XAR             - REAL FORMAT OF INPUT IMAGE
C               XSUM            - REAL FORMAT OF THE SUPERIMPOSED
C                                 SS_GABORED IMAGES
C               XJET            - REAL FORMAT OF THE SUPERIMPOSED
C                                 JET COEFFICIENTS
C               AR_BYTE         - BYTE FORMAT OF THE INPUT IMAGE
C
C       VARIABLES USED:
C
C               NJET_LAYERS     - NUMBER OF JET LAYERS
C               NSIZE           - SIZE OF THE JET LAYER (NXN)
C               NFLG_FILE       - OUTPUT FILE NAME INDICATOR
C               ROW_NUM         - NUMBER OF ROWS
C               COL_NUM         - NUMBER OF COLUMNS
C               NFR             - FLAG TO INDICATE THE STATUS FOR
C                                 READING THE PIXEL LOCATIONS OF
C                                 THE CENTROIDS.  NFR=1 INDICATES
C                                 NOT NECESSARY TO READ THE LOCATIONS
C               NTOT            - TOTAL NUMBER OF LOCATIONS FOUND BY
C                                 ROGGEMANN'S ALGORITHM
C               NX,NY           - LOCATIONS OF THE CENTROIDS (X,Y)
C               RP,CP           - POINTERS ASSOCIATED WITH CENTERING
C                                 THE JETS
C               NH              - HALF OF NSIZE
C               XTOP,RTOP,XT2,  - SAME AS THOSE FOUND IN BSG.FOR
C               XT3,THE_MAX
C
C       WRITTEN BY:  KEVIN W. AYER, CAPT, USAF          DATE:  19 JUN 89
C
C       UPDATED BY:                                     DATE:
C
C
C-------------------------------------------------------------------C


        PROGRAM JETS_MAIN
        DIMENSION       NAR(240,640),NSUM(240,640),NJET(240,640)
        DIMENSION       XAR(240,640),XSUM(240,640),TOP(640)
        DIMENSION       XJET(240,640)
        CHARACTER*24    NAMEIN,NAMEOUT,NAME
        INTEGER         R,C,ROW_NUM,COL_NUM
        BYTE            AR_BYTE(240,640)

C
C  NTOT IS A VALUE OBTAINED BY RUNNING THE SHOW_LAB_FLIR_IM PROGRAM
C

        OPEN(20,FILE='TOT.DAT',STATUS='OLD',FORM='FORMATTED')
```

```
      READ(20,*)NTOT
      CLOSE(20)

      NAME='REFM13.FLR'
      NJET_LAYERS=4
      NSIZE=128

      NFLG_FILE = 0
      ROW_NUM=240
      COL_NUM=640
      NFR=0


      DO L=1,NTOT

C  ZEROING OUT THE TRANSFER ARRAY

      DO I=1,240
         DO J=1,640
            XSUM(I,J)=0.
         END DO
      END DO


      DO 102 I=1,NJET_LAYERS

C  ZEROING OUT THE TRANSFER ARRAY

         DO IN=1,240
            DO JN=1,640
               NJET(IN,JN)=0
            END DO
         END DO

         NH=NSIZE/2

         CALL PIX_LOC(L,NX,NY,NFR,NTOT)

         IF(NY.LT.65.OR.NY.GT.576) THEN
            GOTO 523
         END IF
         TYPE*,NX,NY
         NFR=1
         TYPE*,'AT SIZE',NSIZE,'   NAME OF FILE?'
         ACCEPT 30,NAMEIN

         CALL GET_FLIR_IM(NAMEIN,NAR,240,640)


      RP = NX-NH

      IF(RP.LT.0) THEN
         RP = 1
      END IF

      DO R=1,NSIZE

         CP = NY-NH

         DO C=1,NSIZE
```

```
          NJET(RP,CP)=NAR(RP,CP)

    CP=CP+1

    END DO

RP=RP+1

END DO

RP=NX-NH
DO R=1,NSIZE
    CP=NY-NH
    DO C=1,NSIZE
       XJET(RP,CP)=FLOAT(NJET(RP,CP))
       CP=CP+1
    END DO
RP=RP+1
END DO

RP=NX-NH
DO R=1,NSIZE
    CP=NY-NH
    DO C=1,NSIZE
       XSUM(RP,CP)=XSUM(RP,CP)+XJET(RP,CP)
       CP=CP+1
    END DO
RP=RP+1
END DO



CLOSE(10)


DO C=1,640
    XTOP=0.
    DO R=1,240

          XT2=XSUM(R,C)
          RTOP=XTOP-XT2
           IF(RTOP.LT.0) THEN
           XTOP=XT2
           END IF

    END DO

    TOP(C)=XTOP

END DO

THE_MAX=0.

DO C=1,640
    XT3=TOP(C)
    RTOP=THE_MAX-XT3
    IF(RTOP.LT.0) THEN
    THE_MAX=XT3
    END IF
```

```
        END DO

        RP=NX-NH
        DO R=1,NSIZE
           CP=NY-NH
           DO C=1,NSIZE
              XSUM(RP,CP)=255.*(XSUM(RP,CP)/THE_MAX)
              NSUM(RP,CP)=IFIX(XSUM(RP,CP))
              CP=CP+1
           END DO
        RP=RP+1
        END DO

           NSIZE=NSIZE/2

 102    CONTINUE


           CALL OUT_NAME(NFLG_FILE,NAMEOUT)


           CALL PUT_FLIR_IM(NAMEOUT,NSUM,NX,NY,240,640)

        NSIZE=128
        NFLG_FILE=NFLG_FILE+1
 523    END DO

 30     FORMAT(A24)
        END


C-------------------------------------------------------------------C
C
C       SIMILAR TO SUBROUTINE FOUND IN BSG.FOR ONLY THE NAMES OF THE
C       OUTPUT FILES ARE DIFFERENT.
C
C-------------------------------------------------------------------C


        SUBROUTINE OUT_NAME(NFLG,NAMEOUT)

        CHARACTER*24    NAMEOUT

        IF(NFLG.EQ.0) THEN
           NAMEOUT='JET1.FLR'
        ELSEIF(NFLG.EQ.1) THEN
           NAMEOUT='JET2.FLR'
        ELSEIF(NFLG.EQ.2) THEN
           NAMEOUT='JET3.FLR'
        ELSEIF(NFLG.EQ.3) THEN
           NAMEOUT='JET4.FLR'
        ELSEIF(NFLG.EQ.4) THEN
           NAMEOUT='JET5.FLR'
        ELSEIF(NFLG.EQ.5) THEN
           NAMEOUT='JET6.FLR'
        ELSEIF(NFLG.EQ.6) THEN
           NAMEOUT='JET7.FLR'
        ELSEIF(NFLG.EQ.7) THEN
           NAMEOUT='JET8.FLR'
        ELSEIF(NFLG.EQ.8) THEN
```

```
      NAMEOUT='JET9.FLR'
   ELSEIF(NFLG.EQ.9) THEN
      NAMEOUT='JET10.FLR'
   ELSEIF(NFLG.EQ.10) THEN
      NAMEOUT='JET11.FLR'
   ELSEIF(NFLG.EQ.11) THEN
      NAMEOUT='JET12.FLR'
   ELSEIF(NFLG.EQ.12) THEN
      NAMEOUT='JET13.FLR'
   ELSEIF(NFLG.EQ.13) THEN
      NAMEOUT='JET14.FLR'
   ELSEIF(NFLG.EQ.14) THEN
      NAMEOUT='JET15.FLR'
   ELSEIF(NFLG.EQ.15) THEN
      NAMEOUT='JET16.FLR'
   ELSEIF(NFLG.EQ.16) THEN
      NAMEOUT='JET17.FLR'
   ELSEIF(NFLG.EQ.17) THEN
      NAMEOUT='JET18.FLR'
   ELSEIF(NFLG.EQ.18) THEN
      NAMEOUT='JET19.FLR'
   ELSEIF(NFLG.EQ.19) THEN
      NAMEOUT='JET20.FLR'
   ELSEIF(NFLG.EQ.20) THEN
      NAMEOUT='JET21.FLR'
   ELSEIF(NFLG.EQ.21) THEN
      NAMEOUT='JET22.FLR'
   ELSEIF(NFLG.EQ.22) THEN
      NAMEOUT='JET23.FLR'
   ELSEIF(NFLG.EQ.23) THEN
      NAMEOUT='JET24.FLR'
   ELSEIF(NFLG.EQ.24) THEN
      NAMEOUT='JET25.FLR'
   ELSEIF(NFLG.EQ.25) THEN
      NAMEOUT='JET26.FLR'
   ELSEIF(NFLG.EQ.26) THEN
      NAMEOUT='JET27.FLR'
   ELSEIF(NFLG.EQ.27) THEN
      NAMEOUT='JET28.FLR'
   ELSEIF(NFLG.EQ.28) THEN
      NAMEOUT='JET29.FLR'
   ELSEIF(NFLG.EQ.29) THEN
      NAMEOUT='JET30.FLR'
   ELSEIF(NFLG.EQ.30) THEN
      NAMEOUT='JET31.FLR'

   ELSE
      NAMEOUT='JET32.FLR'
   ENDIF

   RETURN

   END
```

```
C-----------------------------------------------------------------------C
C
C       SAME AS PUT SUBROUTINE FOUND IN BSG.FOR WITH THE FOLLOWING
C       DIFFERENCE:
C
C       ONLY THE PIXEL VALUES AROUND A PARTICULAR JET WILL BE
C       DISPLAYED, AND DISPLAYED IN THE RELATIVE POSITION FOUND
C       IN THE ORIGINAL IMAGE USING THE NX, NY, RP, CP POINTER INDICATORS.
C
C-----------------------------------------------------------------------C

        SUBROUTINE PUT_FLIR_IM(NAMEOUT,NMAG,NX,NY,ROW_NUM,COL_NUM)

        IMPLICIT          INTEGER(A-Z)
        DIMENSION         NMAG(240,640)
        BYTE              NMAG_BYTE(240,640)
        CHARACTER*24      NAMEOUT

        DO R=1,240
           DO C=1,640
              NMAG_BYTE(R,C)=0
           END DO
        END DO


        TYPE*,'WRITE TO FILE: ',NAMEOUT,'START'
        RP=NX-64
        IF(RP.LT.0) THEN
           RP=1
        END IF
        DO R=1,128
              CP=NY-64
           DO C=1,128
              NMAG_BYTE(RP,CP)=MOD(NMAG(RP,CP),256)-128
           CP=CP+1
           END DO
        RP=RP+1
        END DO

        OPEN(20,FILE=NAMEOUT,STATUS='NEW',FORM='UNFORMATTED',
     +  RECL=COL_NUM/4+1,RECORDTYPE='FIXED')

        DO 100 R=1,ROW_NUM
           WRITE(20) (NMAG_BYTE(R,C),C=1,COL_NUM)
100     CONTINUE

        DO R=1,256
           DO C=1,512
              NMAG(R,C)=0
              NMAG_BYTE(R,C)=0
           END DO
        END DO

        CLOSE(20)
        TYPE*,'WRITE TO FILE: ',NAMEOUT,'COMPLETE'
        RETURN
        END
```

```
C-----------------------------------------------------------------C
C
C       THIS SUBROUTINE GIVES PIXEL LOCATIONS PROVIDED BY
C       THE THRESH_ADD_HIST PROGRAM.  PIXEL LOCATIONS ARE
C       READ IN FROM A FILE CALLED LFP.DAT.
C
C       ARRAYS USED:
C
C               NLOCX           - COLUMN LOCATION
C               NLOCY           - ROW LOCATION
C
C       VARIABLES USED:
C
C               I,J             - COORDINATE LOCATION OF CENTROIDS
C
C       ASCII VARIABLES USED:
C
C               NONE
C
C       WRITTEN BY:  K. W. AYER, CAPT, USAF        DATE:  27 JULY 1989
C
C       UPDATED BY:                                DATE:
C
C
C-----------------------------------------------------------------C


        SUBROUTINE PIX_LOC(NLO,NX,NY,NFR,NTOT)

        DIMENSION NLOCX(10000),NLOCY(10000)

        OPEN(10,FILE='LFP.DAT',STATUS='OLD',FORM='FORMATTED')


        IF(NFR.EQ.0) THEN


        DO L=1,NTOT
           READ(10,*)I,J
           NLOCX(L)=I
           NLOCY(L)=J
        END DO
        END IF
        NX=NLOCX(NDO)
        NY=NLOCY(NDO)

        CLOSE(10)
        RETURN
        END

C-----------------------------------------------------------------C
C
C       SAME GET IMAGE SUBROUTINE FOUND IN BSG.FOR
C
C-----------------------------------------------------------------C

      SUBROUTINE GET_FLIR_IM(NAME,AR,ROW_NUM,COL_NUM)
        IMPLICIT INTEGER(A-Z)
```

```
      BYTE              AR_BYTE(240,640)
      DIMENSION         AR(240,640)
      CHARACTER*24      NAME

      OPEN(10,FILE=NAME,STATUS='OLD',FORM='UNFORMATTED',
     +     RECORDTYPE='VARIABLE',ERR=20)
20    OPEN(10,FILE=NAME,STATUS='OLD',FORM='UNFORMATTED',
     +     RECORDTYPE='FIXED')

      TYPE*, 'SUCCESSFUL FILE OPEN'

      DO 100 R = 1,ROW_NUM
            READ(10) (AR_BYTE(R,C),C=1,COL_NUM)
100   CONTINUE

      TYPE*, 'SUCCESSFUL FILE READ'

      DO R = 1,ROW_NUM
            DO C = 1,COL_NUM
                  IF(AR_BYTE(R,C).LT.0)THEN
                  AR(R,C) = AR_BYTE(R,C)+256
                  ELSE
                  AR(R,C) = AR_BYTE(R,C)
                  END IF
            END DO
      END DO
      CLOSE(10)
      RETURN
      END
```

Appendix H. Jet Vector generating FORTRAN source code.

VECT_JETS.FOR

SYNOPSIS: The VECT_JETS program constructs 20 (8) dimensional vectors from the coefficients of the jets made by the JETS program. The vectors are associated with the on center locations determined by the SHOW_LAB_FLIR_IM program.

INPUT: Jet coefficients from the JETS program.

OUTPUT: Vector set matrix for input into MATRIX-X

```
C-----------------------------------------------------------------C
C
C       PROGRAM TO FIND THE VECTOR VERSION OF THE JETS.  EACH PIXEL
C       WITHIN THE 'FOVEAL' REGION, IE: NEAR THE CENTER OF EACH JET
C       CORRESPONDS TO A (NORIENT*NJET_LAYERS) DIMENSIONAL VECTOR.
C       FROM THIS VECTOR, CLASSIFICATION MAY BE ATTEMPTED.
C
C       ARRAYS USED:
C
C               NAR             - INTEGER FORMAT INPUT IMAGE ARRAY
C               XAR             - REAL FORMAT INPUT IMAGE ARRAY
C               NVECT           - VECTOR COMPRISED OF PIXEL VALUES
C                                 FROM EACH LAYER AND ORIENTATION
C                                 ON CENTER
C               RVECT           - REAL FORMAT OF NVECT
C               RMATV           - REAL FORMAT MATRIX FORM OF VECTORS
C
C       VARIABLES USED:
C
C               NJET_LAYERS     - NUMBER OF JET LAYERS
C               NORIENT         - NUMBER OF ORIENTATIONS WITHIN EACH
C                                 LAYER
C               NSIZE           - EXTENT OF LAYER
C               NFLG_FILE       - OUTPUT FILE NAME INDICATOR
C               NFR             - READ/NOREAD LOCATIONS INDICATOR
C               ROW_NUM         - NUMBER OF ROWS
C               COL_NUM         - NUMBER OF COLUMNS
C               NC              - VECTOR ENTRY LOCATION INDICATOR
C               NX,NY           - LOCATIONS OF THE CENTROIDS
C
C       ASCII VARIABLES USED:
C
C               MATNAME         - NAME OF MATRIX CONSTRUCTED OF JET
C                                 VECTORS
C               LOCATION        - NAME OF FILE CONTAINING THE LOCATION
C                                 OF THE CENTER PIXELS OF THE BLOBS
C               TOTAL           - NAME OF THE FILE CONTAINING THE TOTAL
C                                 NUMBER OF VECTORS
C
C
C       WRITTEN BY: KEVIN W. AYER, CAPT, USAF           DATE: 1 AUG 89
C
C       UPDATED BY: KEVIN W. AYER, CAPT, USAF           DATE: 12 AUG 89
C
C
C-----------------------------------------------------------------C


      PROGRAM VECT_MAIN
      DIMENSION       NAR(240,640),XAR(240,640),NVECT(20)
      DOUBLE PRECISION  TOP,RVECT(8),RMATV(8,100)
      CHARACTER*24    NAMEIN,NAMEOUT,NAME,MATNAME,TOTAL,LOCATION
      CHARACTER*24    NAME1,NAME2,NAME3,NAME4,NAME5,NAME6
      CHARACTER*24    NAME7,NAME8,NAME9,NAME10,NAME11,NAME12
      CHARACTER*24    NAME13,NAME14,NAME15,NAME16,NAME17,NAME18
      CHARACTER*24    NAME19,NAME20,NAME21,NAME22,NAME23,NAME24

      INTEGER         R,C,ROW_NUM,COL_NUM
```

```
      OPEN(42,FILE='VECT.DAT',FORM='FORMATTED',STATUS='NEW')
 1    CONTINUE


      TYPE*,'WHAT IS THE NAME OF THE FILE CONTAINING THE TOTAL'
      TYPE*,'NUMBER OF VECTORS GENERATED?'
      ACCEPT 30,TOTAL

      TYPE*,'WHAT IS THE NAME OF THE FILE CONTAINING THE LOCATIONS'
      TYPE*,'OF THE CENTER PIXELS OF THE SEGMENTED BLOBS?'
      ACCEPT 30,LOCATION
C
C  NTOT IS A VALUE OBTAINED BY RUNNING THE SHOW_LAB_FLIR_IM PROGRAM
C
      OPEN(20,FILE=TOTAL,STATUS='OLD',FORM='FORMATTED')
      READ(20,*)NTOT
      CLOSE(20)

      TYPE*,'WHAT DO YOU WISH TO CALL THE MATRIX-X OUTPUT FILE?'
      ACCEPT 30,MATNAME

      type*,'NUMBER OF JET LAYERS?'
      READ(5,*)NJET_LAYERS
      TYPE*,'NUMBER OF ORIENTATIONS PER LAYER?'
      READ(5,*)NORIENT
      TYPE*,'INITIAL GABOR WINDOW SIZE?'
      READ(5,*)NSIZE
      NLOOP=NORIENT*NJET_LAYERS
      NFLG_FILE = 0
      NL=1
      ROW_NUM=240
      COL_NUM=640
      NFR=0
      DUMMY=0.


            TYPE*,'AT SIZE 16, NAME OF FILE?'
            ACCEPT 30,NAME1

            TYPE*,'AT SIZE 16, NAME OF FILE?'
            ACCEPT 30,NAME2

            TYPE*,'AT SIZE 16,  NAME OF FILE?'
            ACCEPT 30,NAME3

            TYPE*,'AT SIZE 16,  NAME OF FILE?'
            ACCEPT 30,NAME4


            TYPE*,'AT SIZE 8, NAME OF FILE?'
            ACCEPT 30,NAME5

            TYPE*,'AT SIZE 8, NAME OF FILE?'
            ACCEPT 30,NAME6

            TYPE*,'AT SIZE 8,  NAME OF FILE?'
            ACCEPT 30,NAME7

            TYPE*,'AT SIZE 8,  NAME OF FILE?'
```

```
      ACCEPT 30,NAME8


DO L=1,NTOT
   DO IN=1,20
        NVECT(IN)=0
   END DO
   NC=1
DO I=1,NJET_LAYERS
   CALL PIX_LOC(LOCATION,L,NX,NY,NFR,NTOT)
   IF(NX.GT.65.OR.NY.GT.576) THEN
      GOTO 523
   END IF
   TYPE*,'ROW    ',NX
   TYPE*,'COLUMN ',NY
   NFR=1
   DO IN=1,NORIENT
      IF(NC.EQ.1) THEN
      NAMEIN=NAME1
      ELSEIF(NC.EQ.2) THEN
      NAMEIN=NAME2
      ELSEIF(NC.EQ.3) THEN
      NAMEIN=NAME3
      ELSEIF(NC.EQ.4) THEN
      NAMEIN=NAME4
      ELSEIF(NC.EQ.5) THEN
      NAMEIN=NAME5
      ELSEIF(NC.EQ.6) THEN
      NAMEIN=NAME6
      ELSEIF(NC.EQ.7) THEN
      NAMEIN=NAME7
      ELSEIF(NC.EQ.8) THEN
      NAMEIN=NAME8
      ELSEIF(NC.EQ.9) THEN
      NAMEIN=NAME9
      ELSEIF(NC.EQ.10) THEN
      NAMEIN=NAME10
      ELSEIF(NC.EQ.11) THEN
      NAMEIN=NAME11
      ELSEIF(NC.EQ.12) THEN
      NAMEIN=NAME12
      ELSEIF(NC.EQ.13) THEN
      NAMEIN=NAME13
      ELSEIF(NC.EQ.14) THEN
      NAMEIN=NAME14
      ELSEIF(NC.EQ.15) THEN
      NAMEIN=NAME15
      ELSEIF(NC.EQ.16) THEN
      NAMEIN=NAME16
      ELSEIF(NC.EQ.17) THEN
      NAMEIN=NAME17
      ELSEIF(NC.EQ.18) THEN
      NAMEIN=NAME18
      ELSEIF(NC.EQ.19) THEN
      NAMEIN=NAME19
      ELSE
      NAMEIN=NAME20
      END IF
      CALL GET_FLIR_IM(NAMEIN,NAR,240,640)
      NVECT(NC)=NAR(NX,NY)
```

```
            RVECT(NC)=DFLOAT(NVECT(NC))

         WRITE(42,*)RVECT(NC)

               NC=NC+1
            END DO
            NSIZE=NSIZE/2
         END DO



C        type*,'DO YOU WISH TO SCALE THE OUTPUT BETWEEN'
C        TYPE*,'ZERO AND ONE, LINEARLY? NO=0, YES=1'
C        READ(5,*)NCHO
C        IF(NCHO.EQ.1) THEN
C        TOP=0.

C        DO IT=1,8
C           DIFF=TOP-RVECT(IT)
C           IF(DIFF.LT.0) THEN
C              TOP=RVECT(IT)
C           END IF
C        END DO
C
C
C        XMIN=RVECT(1)
C
C        DO ITL=2,8
C           XDIFF=XMIN-RVECT(ITL)
C           IF(XDIFF.GT.0) THEN
C              XMIN=RVECT(ITL)
C           END IF
C        END DO
C
C        if(top.eq.xmin)then
C        top=top+xmin
C        endif
C
C        DO IT=1,8
C           RVECT(IT)=(RVECT(IT)-XMIN)/(TOP-XMIN)
C        END DO
C
C        END IF
C


         DO I=1,8
            RMATV(I,NL)=RVECT(I)
         END DO



         NL=NL+1
         NFLG_FILE=NFLG_FILE+1
         NSIZE=128
  523    END DO


         CALL MATSAV(1,MATNAME,8,8,ntot,0,
     +   RMATV,DUMMY,'(1P2E25.17)')
```

```
      TYPE*,'DO YOU WISH TO CONTINUE? NO=0, YES=1'
      READ(5,*)NBO
      IF(NBO.EQ.1) THEN
         GOTO 1
      END IF
      CLOSE(42)
30    FORMAT(A24)
      END



C-----------------------------------------------------------------C
C
C     SAME FORMAT AS PIX_LOC FOUND IN JETS.FOR
C
C-----------------------------------------------------------------C


      SUBROUTINE PIX_LOC(LOCNAME,NDO,NX,NY,NFR,NTOT)

      DIMENSION       NLOCX(10000),NLOCY(10000)
      CHARACTER*24    LOCNAME



      OPEN(10,FILE=LOCNAME,STATUS='OLD',FORM='FORMATTED')


      IF(NFR.EQ.0) THEN


      DO L=1,NTOT
         READ(10,*)I,J
         NLOCX(L)=I
         NLOCY(L)=J
      END DO
      END IF
      NX=NLOCX(NDO)
      NY=NLOCY(NDO)

      CLOSE(10)
      RETURN
      END

C-----------------------------------------------------------------C
C
C     SAME GET IMAGE SUBROUTINE AS FOUND IN PREVIOUS PROGRAMS
C
C-----------------------------------------------------------------C

     SUBROUTINE GET_FLIR_IM(NAME,AR,ROW_NUM,COL_NUM)
      IMPLICIT INTEGER(A-Z)
      BYTE            AR_BYTE(240,640)
      DIMENSION       AR(240,640)
      CHARACTER*24    NAME

      OPEN(10,FILE=NAME,STATUS='OLD',FORM='UNFORMATTED',
     +     RECORDTYPE='VARIABLE',ERR=20)
20    OPEN(10,FILE=NAME,STATUS='OLD',FORM='UNFORMATTED',
```

```
     +       RECORDTYPE='FIXED')

        TYPE*, 'SUCCESSFUL FILE OPEN'

        DO 100 R = 1,ROW_NUM
               READ(10)(AR_BYTE(R,C),C=1,COL_NUM)
 100    CONTINUE

        TYPE*, 'SUCCESSFUL FILE READ'

        DO R = 1,ROW_NUM
               DO C = 1,COL_NUM
                      IF(AR_BYTE(R,C).LT.0)THEN
                      AR(R,C) = AR_BYTE(R,C)+256
                      ELSE
                      AR(R,C) = AR_BYTE(R,C)
                      END IF
               END DO
        END DO
        CLOSE(10)
        RETURN
        END




      SUBROUTINE MATSAV ( LUNIT, NAME, NR, M, N, IMG,
     $                    XREAL, XIMAG, FORMT )
C
C ------------------------------------------------------------------------
C |            MATRIXx TM Software V7.0    (C) Copyright 1988            |
C |           INTEGRATED SYSTEMS INC., Santa Clara, California            |
C |                                                                       |
C | Published Work.  Permission is granted to any individual or           |
C | institution to modify, copy or use this subroutine except for         |
C | explicitly commercial purposes.                                       |
C ------------------------------------------------------------------------
C |                                                                       |
C | MATSAV writes a matrix to a file in a MATRIXx-readable format.  It    |
C | can be called by programs that want to write data to MATRIXx.         |
C |                                                                       |
C | Files written with MATSAV can be read into MATRIXx with the LOAD      |
C | command.                                                              |
C |                                                                       |
C |-----------------------------------------------------------------------|
C | Param. | Type           | On input-                          | On output- |
C |-----------------------------------------------------------------------|
C |        |                |                                    |            |
C | LUNIT  | INTEGER        | Fortran logical unit number.       | unchanged. |
C |        |                |                                    |            |
C | NAME   | CHARACTER*(*)  | Name of the matrix.  One al-       | unchanged. |
C |        | (maximum       | phabetic followed by up to 9       |            |
C |        |  length 10)    | alphanumeric characters.           |            |
C |        |                |                                    |            |
C | NR     | INTEGER        | Row-dimension in the               | unchanged. |
C |        |                | defining dimension or type         |            |
C |        |                | statement in the calling           |            |
```

		program. NR must be greater than or equal to M.	
M	INTEGER	Number of rows of the matrix	unchanged.
N	INTEGER	Number of columns of the matrix.	unchanged.
IMG	INTEGER	If IMG = 0, the imaginary part (XIMAG) is assumed to be zero and is not saved.	unchanged.
XREAL	DOUBLE PRECISION	Real part of the matrix to be saved.	unchanged.
XIMAG	DOUBLE PRECISION	Imaginary part of the matrix to be saved.	unchanged.
FORMT	CHARACTER*(*) (maximum length 20)	String containing the Fortran format to be used for writing the elements of the matrix. For machine-independent files use '(1P2E25.17)' For machine-dependent (fast, compact) files use '(10A8)' or '(10Z16)'	unchanged.

Example: The following Fortran program generates a simple identity matrix in X and writes it to Fortran unit 1. Assume that unit 1 has been preallocated as file (data set) TEST.

```
      DOUBLE PRECISION X(20,3), DUMMY
        DO 200 J=1,3
          DO 100 I=1,10
            X(I,J)=0.0D0
 100      CONTINUE
          X(J,J)=1.0D0
 200    CONTINUE

      CALL MATSAV( 1, 'AMATRIX', 20, 10, 3, 0,
     $             X, DUMMY, '(1P2E25.17)' )

      STOP
      END
```

After this program runs, invoke MATRIXx and type:

```
<> LOAD 'TEST'
```

This will put X on the stack as a variable named AMATRIX.

When the objective is to convert a file written by another software package into a MATRIXx-readable file, you will have to write a small program that reads the data in using the format of the

```
C | other software package then writes it out by calling MATSAV.       |
C ---------------------------------------------------------------------
C
      INTEGER             LUNIT, M, N, NR, IMG
      CHARACTER*(*)       NAME, FORMT
      DOUBLE PRECISION              XREAL(NR,*), XIMAG(NR,*)
      CHARACTER           NAM*10, FORM*20
C
C                                 ---------------------------------------
C                                 | write header record.                |
C                                 ---------------------------------------
      NAM=NAME
      FORM=FORMT
      OPEN(LUNIT,FILE=NAM,STATUS='NEW')

      WRITE(LUNIT,'(A10,3I5,A20)') NAM,M,N,IMG,FORM
C                                 ---------------------------------------
C                                 | write real-part of the matrix.       |
C                                 ---------------------------------------
      WRITE(LUNIT,FORM) ((XREAL(I,J),I=1,M),J=1,N)
C                                 ---------------------------------------
C                                 | write imaginary-part if nonzero.     |
C                                 ---------------------------------------
      IF(IMG.NE.0) WRITE(LUNIT,FORM) ((XIMAG(I,J),I=1,M),J=1,N)

      CLOSE(LUNIT)
      RETURN
      END
```

Appendix I. MATRIX-X code for generating the Minkowski-2 Space Distances.

CORR.MXE

SYNOPSIS: The CORR.MXE program is a MATRIX-X readable code that calculates the Minkowski two, or Euclidean, distance between vectors generated by the VECT_JETS program. Each vector from a given image is put together in such a way as to make a matrix X by 20 (or X by 8). An exemplar matrix is used which is comprised of one of the vector sets generated by the VECT_JETS program. Each vector is then compared to the exemplar set using the Minkowski two space distance rule.

EXAMPLE: Let the exemplar vector set make a 7 by 8 matrix, seven vectors each having 8 components. A test vector set makes a 14 by 8 matrix, 14 vectors each having 8 components. The resulting distance matrix will be 14 by 7, where each component of the distance matrix is the compared distance between vectors of the exemplar set and the test vector set.

INPUT: Vector sets converted into MATRIX-X format.

OUTPUT: Distance matrix between the exemplar vector set and test vector set.

```
*****************************************************************
*
*
*   CORR.MXE.  MATRIX-X code for calculating the Euclidean
*
*   distance between vectors generated from the VECT_JETS program.
*
*
*   ARRAYS USED:
*
*
*
*          C(i,k)          - Matrix of distances
*          A(i,j)          - Matrix made from test vectors
*          B(j,k)          - Matrix made from exemplar vectors
*
*   VARIABLES USED:
*
*
*
*          rows            - Number of rows associated with test
*                            matrix
*          cols            - Number of cols associated with
*                            exemplar matrix
*
*
*   ASCII VARIABLES USED:
*
*
*
*          None
*
*
*
*   Written by:  Kevin W. Ayer. Capt, USAF     DATE:  14 Aug 89
*
*
*   Updated by:                                DATE:
*
*
*
*****************************************************************
```

The following is the basic code for vector comparison. For specific code, refer to CORR1.MXE or CORR2.MXE.

```
FOR i = 1 : rows ;...

    FOR k = 1 : cols ;...

        C(i,k) = 0 ;...

                FOR j = 1 : 8 ;...

                    C(i,k) = C(i,k) + ( A(i,j) - B(j,k) ) ** 2 ;...

                END ;...
         C(I,k) = SQRT ( C(i,k) ) ;...

    END ;...
END ;...
```

[NOTE: MATRK24(j,k), MATRM13(j,k) or MATRM14(j,k) is substituted for B(j,k) depending on the data set used. CORR1.MXE uses MATRM19(j,k) and CORR2.MXE uses MATRK24(j,k) as template vector sets]

```
C | other software package then writes it out by calling MATSAV.       |
C ---------------------------------------------------------------------
C
      INTEGER             LUNIT, M, N, NR, IMG
      CHARACTER*(*)       NAME, FORMT
      DOUBLE PRECISION              XREAL(NR,*), XIMAG(NR,*)
      CHARACTER           NAM*10, FORM*20
C
C                                 ---------------------------------------
C                                 | write header record.                |
C                                 ---------------------------------------
      NAM=NAME
      FORM=FORMT
      OPEN(LUNIT,FILE=NAM,STATUS='NEW')

      WRITE(LUNIT,'(A10,3I5,A20)') NAM,M,N,IMG,FORM
C                                 ---------------------------------------
C                                 | write real-part of the matrix.       |
C                                 ---------------------------------------
      WRITE(LUNIT,FORM) ((XREAL(I,J),I=1,M),J=1,N)
C                                 ---------------------------------------
C                                 | write imaginary-part if nonzero.     |
C                                 ---------------------------------------
      IF(IMG.NE.0) WRITE(LUNIT,FORM) ((XIMAG(I,J),I=1,M),J=1,N)

      CLOSE(LUNIT)
      RETURN
      END
```

Appendix I. MATRIX-X code for generating the Minkowski-2 Space Distances.

CORR.MXE

SYNOPSIS: The CORR.MXE program is a MATRIX-X readable code that calculates the Minkowski two, or Euclidean, distance between vectors generated by the VECT_JETS program. Each vector from a given image is put together in such a way as to make a matrix X by 20 (or X by 8). An exemplar matrix is used which is comprised of one of the vector sets generated by the VECT_JETS program. Each vector is then compared to the exemplar set using the Minkowski two space distance rule.

EXAMPLE: Let the exemplar vector set make a 7 by 8 matrix, seven vectors each having 8 components. A test vector set makes a 14 by 8 matrix, 14 vectors each having 8 components. The resulting distance matrix will be 14 by 7, where each component of the distance matrix is the compared distance between vectors of the exemplar set and the test vector set.

INPUT: Vector sets converted into MATRIX-X format.

OUTPUT: Distance matrix between the exemplar vector set and test vector set.

```
*****************************************************************
*
*
*   CORR.MXE.  MATRIX-X code for calculating the Euclidean
*
*   distance between vectors generated from the VECT_JETS program.
*
*
*   ARRAYS USED:
*
*
*
*          C(i,k)          - Matrix of distances
*          A(i,j)          - Matrix made from test vectors
*          B(j,k)          - Matrix made from exemplar vectors
*
*   VARIABLES USED:
*
*
*
*          rows            - Number of rows associated with test
*                            matrix
*          cols            - Number of cols associated with
*                            exemplar matrix
*
*
*   ASCII VARIABLES USED:
*
*
*
*          None
*
*
*
*   Written by:  Kevin W. Ayer. Capt, USAF     DATE:  14 Aug 89
*
*
*   Updated by:                                DATE:
*
*
*
*****************************************************************
```

The following is the basic code for vector comparison. For specific code, refer to CORR1.MXE or CORR2.MXE.

```
FOR i = 1 : rows ;...

    FOR k = 1 : cols ;...

        C(i,k) = 0 ;...

                FOR j = 1 : 8 ;...

                    C(i,k) = C(i,k) + ( A(i,j) - B(j,k) ) ** 2 ;...

                END ;...
         C(I,k) = SQRT ( C(i,k) ) ;...

    END ;...
END ;...
```

[NOTE: MATRK24(j,k), MATRM13(j,k) or MATRM14(j,k) is substituted for B(j,k) depending on the data set used. CORR1.MXE uses MATRM19(j,k) and CORR2.MXE uses MATRK24(j,k) as template vector sets]

```
FOR I=1:ROWS;...
  FOR K=1:COLS;...
    C(I,K)=0;...
    FOR J=1:8;...
      C(I,K)=C(I,K)+(A(I,J)-MATRJ19(J,K))**2;...
    END;...
  C(I,K)=SQRT(C(I,K));...
  END;...
END;...
NUM=7;...
FOR L=1:6;...
  FOR I=1:ROWS;...
  MIN(L,I)=5.;...
    FOR K=1:COLS;...
    DIFF= MIN(L,I) - C(I,K);...
        IF DIFF>0,MIN(L,I)=C(I,K);END;...
        IF DIFF>0,LOC=K;END;...
    END;...
  C(I,LOC)=100.;...
  MIN(NUM,I)=LOC;...
  END;...
NUM=NUM+1;...
END;...
FOR I=2:6;...
  IM1=I-1;...
  FOR J=1:ROWS;...
    FOM(IM1,J)=MIN(I,J)/MIN(1,J);...
  END;...
END;...
```

```
FOR I=1:ROWS;...
  FOR K=1:COLS;...
    C(I,K)=0;...
    FOR J=1:8;...
      C(I,K)=C(I,K)+(A(I,J)-MATRM13(J,K))**2;...
    END;...
  C(I,K)=SQRT(C(I,K));...
  END;...
END;...
NUM=7;...
FOR L=1:6;...
  FOR I=1:ROWS;...
  MIN(L,I)=5.;...
    FOR K=1:COLS;...
    DIFF= MIN(L,I) - C(I,K);...
        IF DIFF>0,MIN(L,I)=C(I,K);END;...
        IF DIFF>0,LOC=K;END;...
    END;...
  C(I,LOC)=100.;...
  MIN(NUM,I)=LOC;...
  END;...
NUM=NUM+1;...
END;...
FOR I=2:6;...
  IM1=I-1;...
  FOR J=1:ROWS;...
    FOM(IM1,J)=MIN(I,J)/MIN(1,J);...
  END;...
END;...
```

Appendix J. Making a hard copy of displayed images.

Paper copies of Gabor filtered images are produced on the SUN 4.0 WorkStation computer system using the gray scale technique. The gray scale technique involved converting the pixel values of the images generated on the VAX Workstation to 0 or 255 using a program called HTONE2, an ADA coded program provided by Capt Dennis Ruck. Each image file had to be converted to a standard binary format using the VAX system command CONVERT/FDL=filename. The filename is an image file with specific characteristics which remove all record specific parameters such as record length, record size, carriage controls, etc., prior to transfer to the UNIX based Sun 4.0 Workstation. The standard binary files are transferred from the VAXStation-3 system to the SUN 4.0 Color Workstation system. Note that the SUN 4.0 Workstation MUST be a Color Station for the program to work. Once the image files are transferred to the Sun 4.0 Workstation, the images are converted to a PIXRECT format for image display on the screen. The screen display of the images is then converted to a raster file for transfer using a capture routine provided by Capt Dennis Ruck. The capture routine requires known x and y extent of the PIXRECT converted images. The resulting rastered images are then sent to a laser printer for hard copy.

An alternative method for generating images uses a binary to TIFF conversion routine called FLR2TIF, written in C-Language code by Capt John Cline. The images created on the VAXStation-3 system are converted in a similar fashion as the Ruck algorithm and sent to the SUN 4.0 system. The images are, then, transferred to a personal computer (PC, Z-248) for conversion from binary to TIFF. FLR2TIF produces an 8 bit grey scale image as opposed to 4 bit (HTONE2).

Appendix K. Schematic of Gabor Based Segmentation and Classification Program.

Legend:

Dotted line boxes represent separate programs.

Solid boxes represent decision nodes.

Octagons represent input nodes.

Diamonds represent output nodes.

Input raw FLIR image consisting of 240 by 640 pixels.

Decision requested for classification or segmentation.

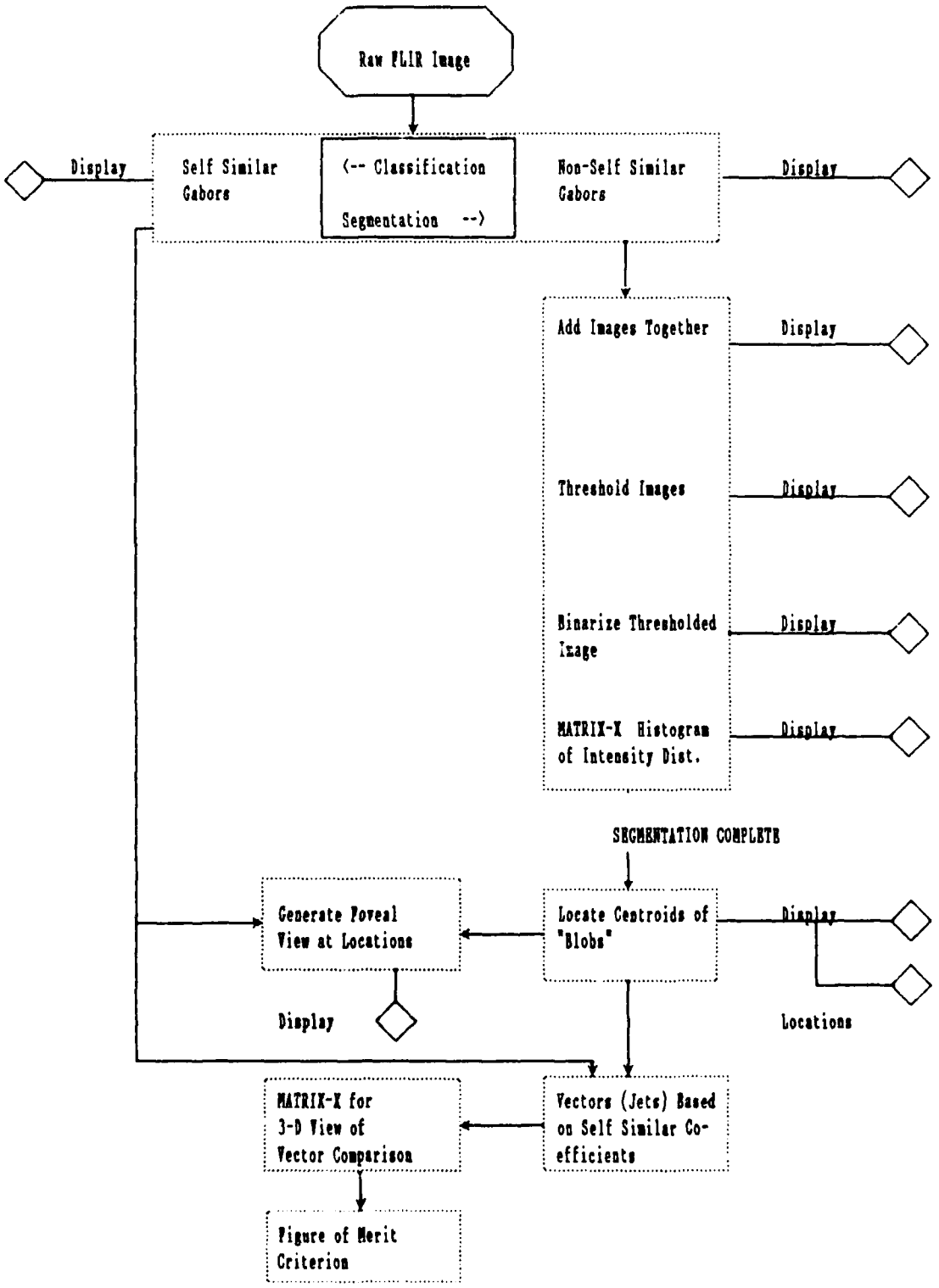
If classification is requested, program executes self similar Gabor function subroutine. Screen display of self similar Gabor transformed FLIR image is an option at this point in the program.

If segmentation is requested, program executes non-self similar Gabor function subroutine. Screen display of non-self similar Gabor transformed FLIR image is an option at this point in the program.

Segmentation. Once the non-self similar Gabor filtered images are generated, they are processed by an algorithm which superimposes them. The superimposed images are histogrammed based on the individual pixel values, which correspond to image intensity. A threshold is established based on the intensity distribution (histogram). Segmentation is considered complete at this point.

Classification. The self similar Gabor transformed images establish the data set from which the Gabor coefficients for foveal views are generated. They also comprise the data set from which the jet vectors will be generated.

The segmented, thresholded versions of the FLIR images are processed by an algorithm that finds the centroids of the segmented "blobs". From these centroid locations, jet vectors will be constructed. The vectors (jets) are concatenated into a MATRIX-X readable file for test to exemplar comparison. Classification is completed at this point.



CLASSIFICATION OF TARGETS

Appendix L. Equipment and resource listing.

1. Computer Resources.

a. Sun 4.0 WorkStations UNIX Computer Systems.

1) Graphics Sun WorkStation.

AFIT, Bldg. 640, Room 243.

2) Array Multiplier Sun WorkStation.

AFIT, Bldg. 640, Room 243.

b. VAX VMS Computer Systems.

AFIT, Bldg. 642, Signal Processing Lab (Rm. 2011)

c. Personal Computer (PC) Resources.

Zenith 248 PC computers.

AFIT, Bldg. 640, Room 243.

d. AFIT school computers (Names are acronyms for the mainframe computers available at AFIT).

1) CSC.

2) ICC.

2. Other resources.

Human Bioengineering Laboratories, Bldg. 248, Wright-Patterson AFB, OH, Area B., contact: Dr. Mark Cannon and Mr. Steve Fullencamp.

Appendix M. Increasing the sinusoidal variation under the self similar Gaussian Cofactor.

A novel approach for combining the segmentation capabilities of non-self similar Gabor functions with the ease of generating dilating Gabor functions (self similar) is presented. This approach introduces the notion of a user definable number of cycles under the Gaussian envelope coupled with dilating the function via changing the Gabor window size. Care must be taken to ensure near zero Gabor

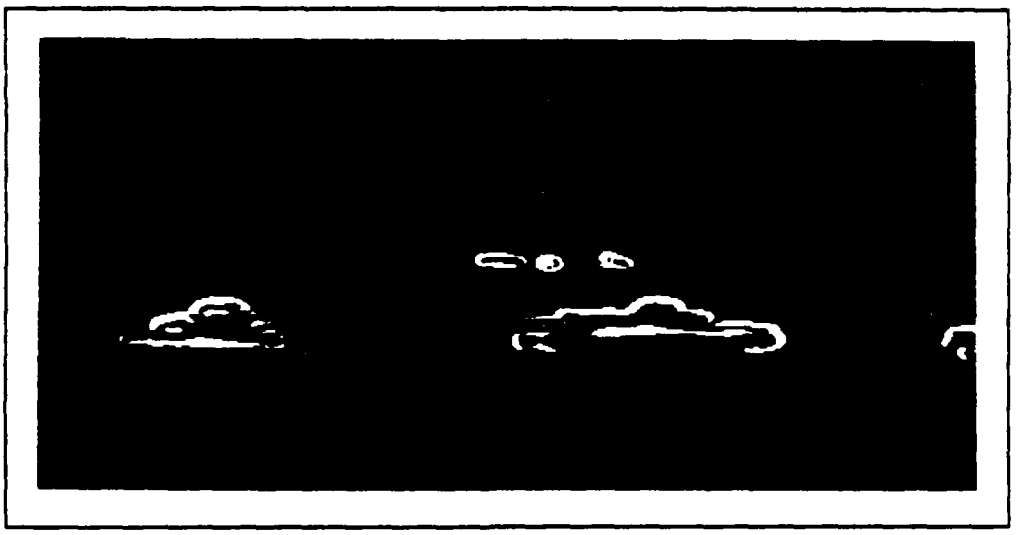


Figure AM-1. Effect of increasing the number of cycles under the Gaussian envelope of a self similar Gabor function. Binarized version of modified Gabor transformed image. Note the complete boundaries around the targets.

function values at the maximum extent of the Gabor window, thereby reducing the effect of Gabor function windowing. Figure AM-1 shows the effects of increasing the number of cycles under the self similar Gabor function's Gaussian envelope. As is clearly evident in Figure AM-1, by modifying the traditional "one cycle" self similar Gabor function, segmentation is possible. Note the complete to near complete boundaries around the targets within the Gabor processed FLIR image.

Appendix N. Grossberg's Boundary Contour-Feature Contour System (8:87-116).

In a series of papers, Grossberg describes an interactive system that establishes boundaries around an object of interest, then completes the region surrounded by the boundaries.

This hybrid system is referred to as the Boundary Contour-Feature Contour System (BC-FC system). Through the interaction of the boundary contour system with the feature contour system, boundary completion and segmentation occur earlier in image processing.

The output of the boundary contour system stage are fill-in generator (FIGs) and fill-in barriers (FIBs) which feed into parallel feature contour system stages: a monocular and a binocular stage.

The monocular feature contour stage suppresses prohibitive or inconsistent monocular color and brightness signals while the binocular FC stage performs a similar task for inconsistent binocular color and brightness signals. The signals which survive this multiple filtering stage act to fill in only those regions possessing monocular and binocular consistent signals.

Grossberg's monocular and binocular theories, based on the boundary contour-feature contour system, model how chromatically

broadband cortical complex cells adaptively tune to process information about position, orientation, spatial frequency, positional disparity, and orientational disparity.

Appendix O. Lambert preprocessing of FLIR images (15).

The primary purpose of "Lambertizing" an image is to eliminate the dependency most processes have on image intensity. The Gabor transform is not immune to image intensities. Therefore, in an effort to eliminate the image intensity dependency, the raw FLIR images were subjected to "Lambertization".

Lambertization involves computing the local average intensity about a central location, subtracting the local average intensity from the central location intensity, then adding this computed value to a set intensity value

$$\text{New Pixel Value} = 128 + (\text{Center Pixel Value} - \text{Local Average}) \quad (16)$$

Note that the new pixel value will be different from 128 by the difference between the center pixel value and its corresponding local average. Hence the resulting image will be independent of global and local intensities, comprised of pixel intensity values deviating about a set pixel value, 128.

Figure AO-1 shows the effect of "Lambertizing" FLIR image REFK41.FLR. As is evident in figure AO-1, "Lambertizing" the FLIR image introduces severe corruption of the image. However, this corruption is not detrimental to the Gabor transformation process.

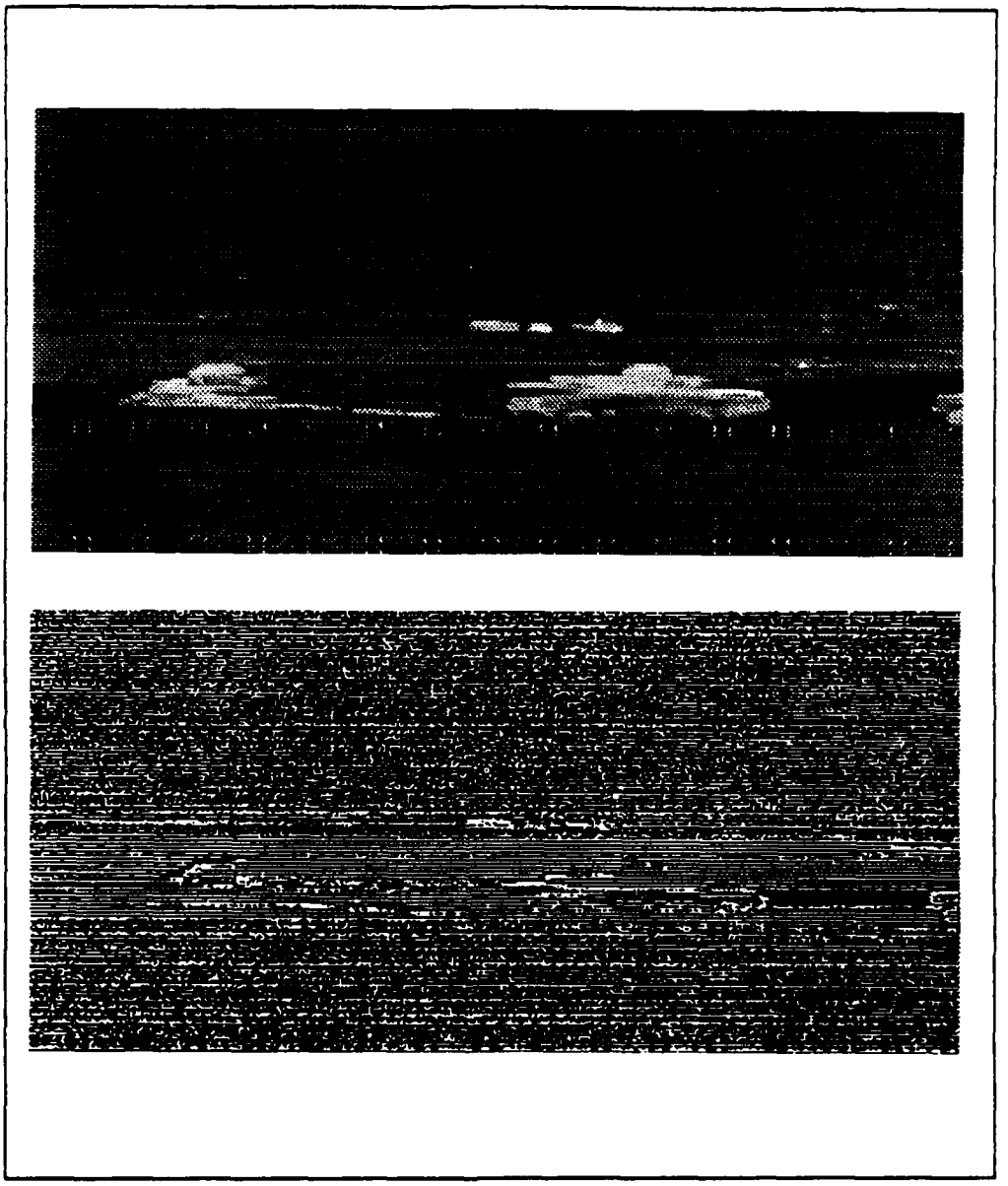


Figure AO-1. Effects of "Lambertizing" FLIR image REFK41.FLR. The top image is the original FLIR image. The bottom image is the Lambert version. Note the clutter introduced by Lambertization.

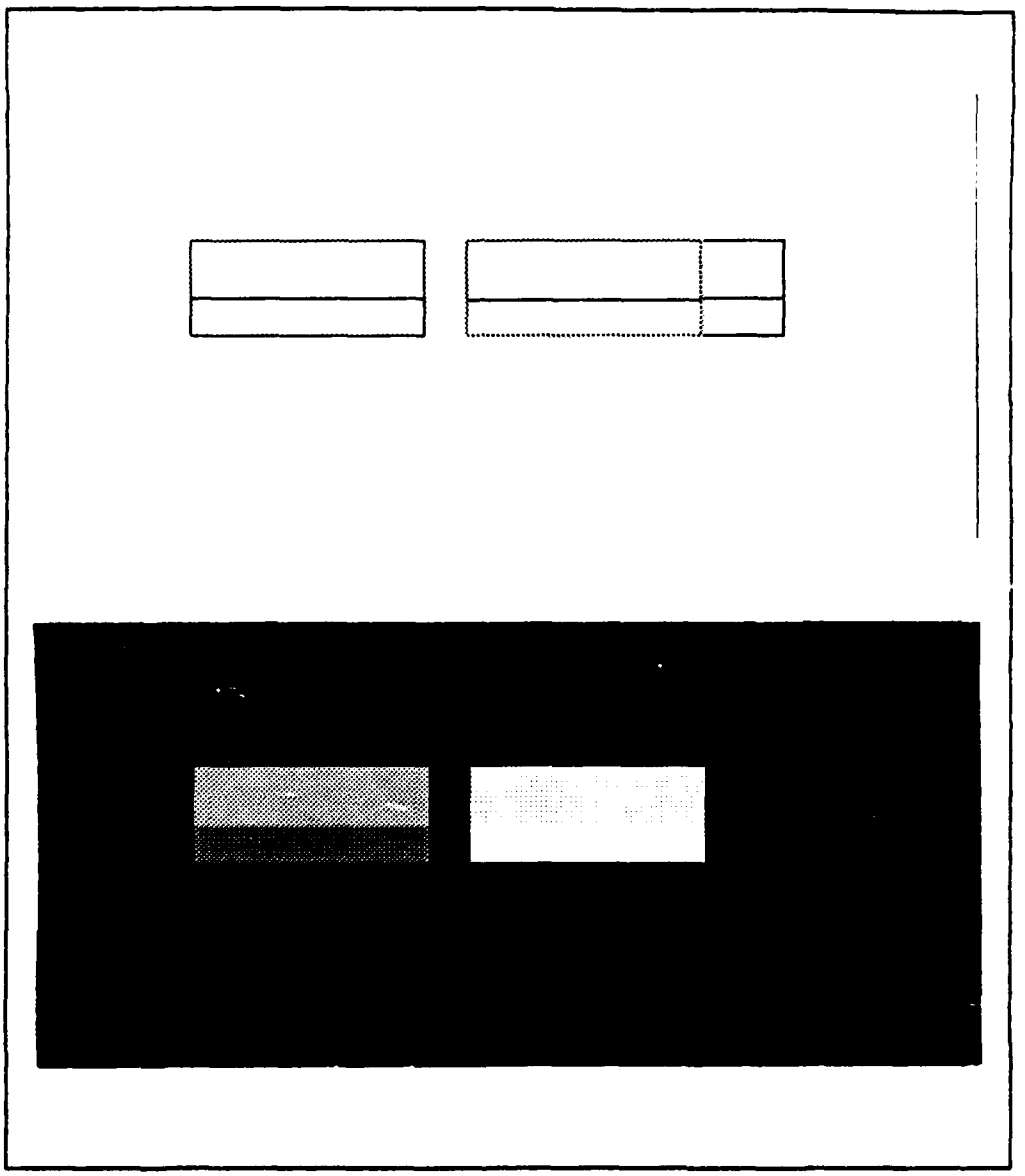


Figure AO-3. a. Lambertization of test image. Outlining of rectangles within original test image. b. Original test image, RECT_TEST.DAT. Lambertization provides indication of boundaries between low intensity regions.

process distinguishes the boundary between the background (pixel value: 0) and the rectangle (pixel value: 10).

The automatic thresholding algorithm developed for this research was designed for Gabor transformation and not for preprocessed, "Lambertized", Gabor transformation.

Therefore, the thresholding process will produce more distinct regions of interest for non-preprocessed Gabor transformed images than for Lambertized Gabor transformed images (Figures AO-4a and b). As can be seen in figure AO-4a, the edges of the targets within the FLIR image are more distinct when no preprocessing is performed. These results are typical of processing the FLIR images using only Gabor transformation techniques versus those using preprocessing, "Lambertizing", techniques.

Figure AO-2 shows the effects of Gabor transforming the "Lambertized" version of FLIR image REFK41.FLR. The results of "Lambertization" are inconclusive using the image set for this research primarily because of the lack of dim targets in this set of images. Lambertization works well for images possessing objects

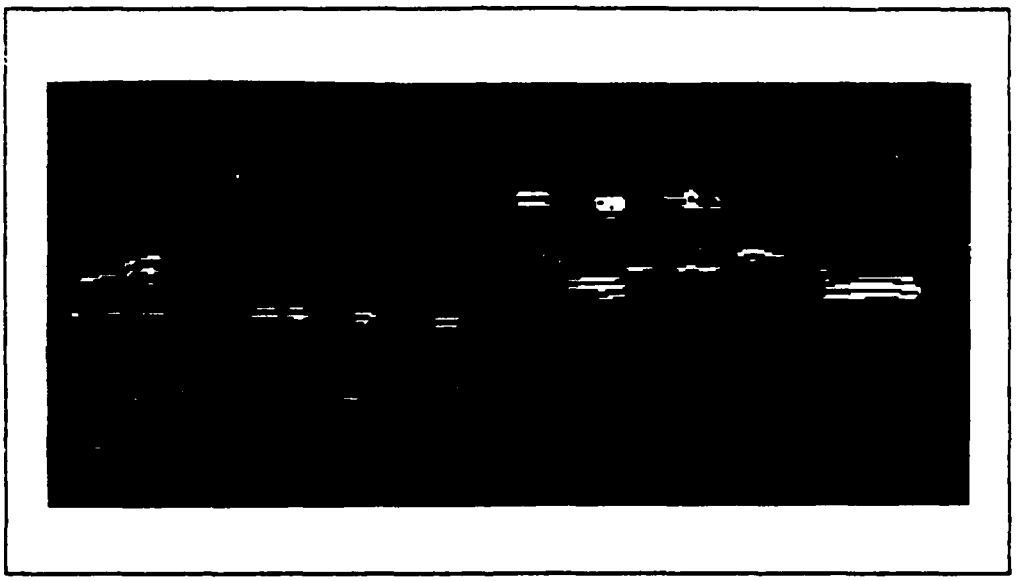


Figure AO-2. Gabor transformation of "Lambertized" version of FLIR image REFK41.FLR (frequency: 3, orientations: 0 and 90 degrees). Lambertization reduces the effectiveness of Gabor transforms used for this thesis.

with widely varying illumination (Figures AO-3a and b). The rectangles along the right side of the second block of rectangles are barely detectable in this image. However, the Lambertization

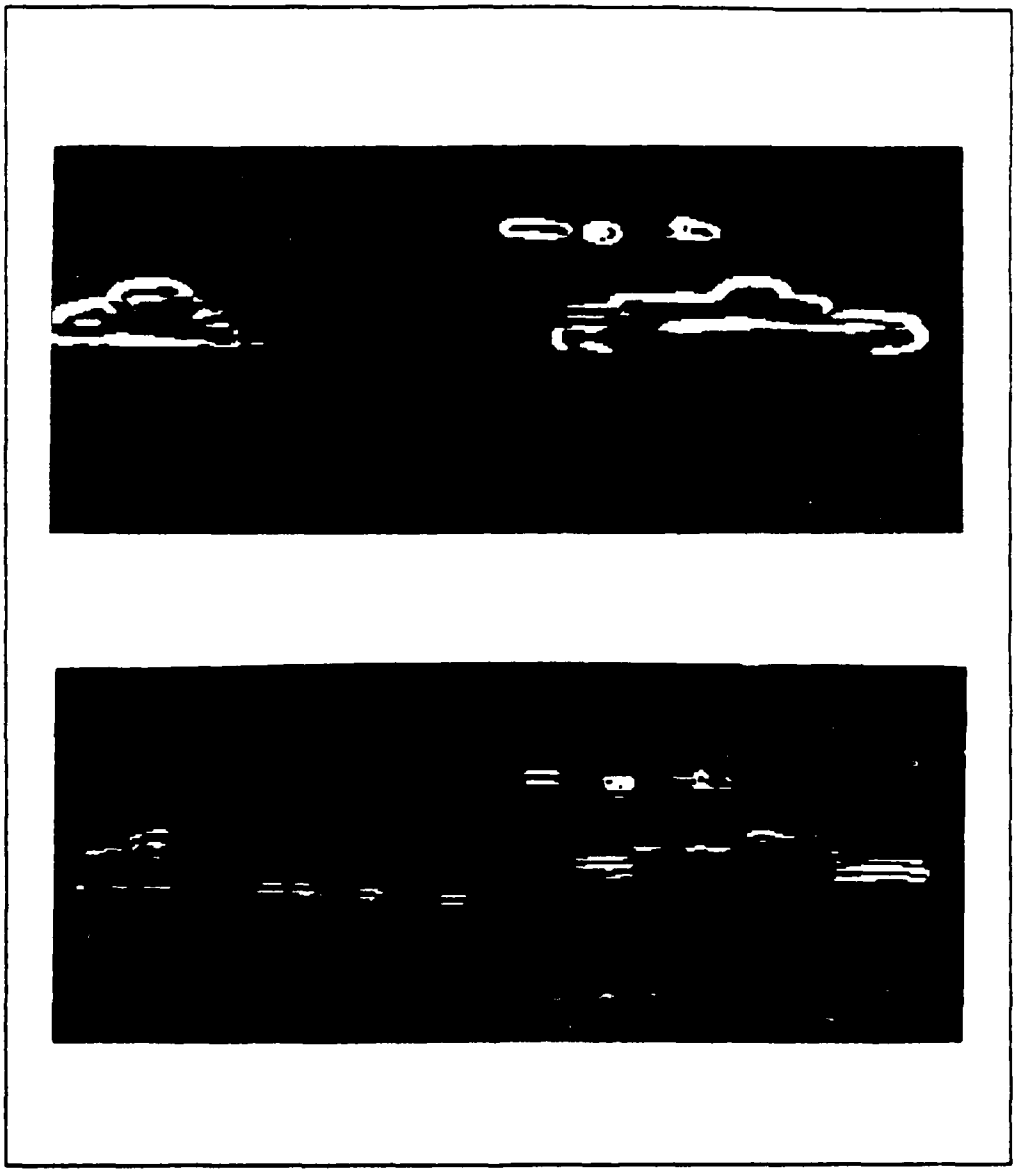


Figure AO-4. a. Gabor transformation of FLIR image REFK41.FLR (No Lambertization). b. Gabor transformation of FLIR image REFK41.FLR (After Lambertization). Lambertization reduces effectiveness of Gabor transformation used in this thesis.

VITA

Captain Kevin W. Ayer [redacted] in 1978 and attended the University of Texas at El Paso, from which he received the degree of Bachelor of Science in Chemistry, with Honors, in 1982. Upon graduation, he was employed by the Department of Metallurgical Engineering, UTEP. In March 1984, he entered the United States Air Force, receiving his commission from Officer Training School in May 1984. After receiving his commission, he attended Auburn University where he received the degree of Bachelor of Science Electrical Engineering in March 1986. He then served as the Director of Reliability and Maintainability, Propulsion Division, Tinker AFB, OK until entering the School of Engineering, Air Force Institute of Technology, in May 1988.

[redacted]

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GEO/ENG/89D-01	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Gabor Transforms for Forward Looking Infrared Image Segmentation

12. PERSONAL AUTHOR(S)
Kevin W. Ayer, Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1989 December	221

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Image Processing, Gabor Transforms, FLIR imagery Segmentation

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Advisor: Steven K. Rogers, MAJ, USAF
Associate Professor of Electrical Engineering

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Steven K. Rogers, MAJ, USAF	(513) 255-9266	AFIT/ENG

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

19 Continued:

This research investigated the difficult task of segmenting targets from non-targets in cluttered FLIR images. The primary means for segmenting targets was the biologically motivated use of Gabor Transforms. This research successfully produced a complete segmentation system based on the correlations between FLIR images and non-self similar or modified self similar Gabor functions. A feasibility study into the use of Gabor correlation coefficients for target classification was performed. An optical design is introduced for computing the Gabor coefficients and correlations.